Efficient Bootstrapping in Fully Homomorphic Encryption for Matrix Arithmetic

Eric Crockett[1], Craig Gentry[1], Hyojun Kim[2], Yeongmin Lee[2], and Yongwoo Lee[2,3]

[1] Cornami {ecrockett, cgentry}@cornami.com
[2] DESILO Inc. {hyojun.kim, yeongmin.lee}@desilo.ai
[3] Inha University yongwoo@inha.ac.kr

Abstract. Recently, Gentry and Lee (GL) proposed a fully homomorphic encryption (FHE) scheme optimized for matrix arithmetic. In this paper, we propose an efficient bootstrapping technique for the GL scheme. Our core idea leverages the linearity of the slot–coefficient transformations, namely CtS and StC: we formulate these operations as ciphertext–plaintext matrix multiplications, which are natively supported by the GL scheme. As a result, the proposed method reduces the number of key-switching operations per step to a small constant. To enable this, we first generalize the GL scheme to matrices of non-power-of-two dimensions by introducing a generalized definition of the trace over commutative rings and proving that it commutes with decryption. Our bootstrapping adopts the CKKS paradigm: ModRaise, CtS, EvalMod, and StC. Typically, CtS/StC and EvalMod dominate runtime and depth, respectively; our optimization shifts the bottleneck to EvalMod for both. A proof-of-concept implementation shows that linear transformations account for 20.1% of the total bootstrapping time, compared to 54.9–71.7% in prior CKKS bootstrapping, and that, despite lacking low-level optimizations, our amortized CtS runtime is still about 3× faster than the well-optimized library (Lattigo).

Keywords: Bootstrapping · Fully Homomorphic Encryption (FHE) · Matrix Multiplication · Ring-Learning with Errors (RLWE).

1 Introduction

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that allows computation on encrypted data without decryption. Starting from Gentry's breakthrough result [22, 23], FHE research has grown rapidly. The SIMD (Single Instruction Multiple Data) packing technique is one of the most important features of modern FHE [25], allowing parallel computation with single ciphertext. The Brakerski-Gentry-Vaikuntanathan (BGV) [12] and Brakerski/Fan-Vercauteren (BFV) [11, 20] schemes are designed for SIMD operations over $\mathbb{Z}_t$, while the Cheon-Kim-Kim-Song (CKKS) [16] scheme is designed for SIMD operations over $\mathbb{C}$. Using the Chinese Remainder Theorem (CRT), we can construct a

decoding homomorphism from polynomials to vectors, i.e., $\mathbb{Z}_t[X]/(X^N+1) \cong \mathbb{Z}_t^N$ or via the canonical embedding $\mathbb{Z}[X]/(X^N+1) \to \mathbb{C}^{N/2}$. We refer to the elements of this mapped space as *slots*.

Recently, Gentry and Lee (GL) [27] proposed a new FHE scheme for matrix arithmetic. The GL scheme introduces a novel encoding method that supports matrix multiplication, Hadamard products, and addition without requiring inter-slot transformations. Unlike prior SIMD FHE schemes, which primarily focus on component-wise addition and multiplication induced by the decoding homomorphism, the GL scheme exploits the linearity of the trace map and constructs the decryption function to commute with the trace operation. Crucially, the trace corresponds to matrix multiplication over the slots, yet it can be evaluated via standard matrix multiplication of the ciphertext coefficients in $\mathbb{Z}_q$.

Modern FHE schemes, including BFV, BGV, CKKS, and GL, are based on the Ring Learning with Errors (RLWE) problem, which inherently introduces noise during encryption. Consequently, this noise accumulates with each homomorphic computation. When the noise magnitude exceeds the certain threshold, decryption fails; such schemes are therefore termed *leveled* homomorphic encryption. To enable unbounded computation depth, the decryption circuit is evaluated homomorphically—a process known as bootstrapping [22].

Current bootstrapping techniques for SIMD FHE schemes typically rely on linear transformations between the coefficient and slot representations [24]. The process generally begins with a homomorphic encoding step, known as CtS, which maps the coefficients of the noisy ciphertext into the plaintext slots. Subsequently, the decryption circuit is evaluated homomorphically to reduce the noise, followed by the StC transformation that restores the original slot representation. Research on CKKS bootstrapping has garnered significant attention due to its efficiency and broad applicability. Since its initial proposal by Cheon et al. [14], significant efforts have been made to optimize the linear transformations — coefficients to slots (CtS) and slots to coefficients (StC) [17, 13, 9, 31, 6]. Parallel improvements to the homomorphic decryption procedure have been introduced in [13, 35, 10, 4]. Furthermore, CKKS bootstrapping has been adapted for other HE schemes or extended to support new functionalities [7, 33, 3].

While existing bootstrapping methods are compatible with the GL scheme, its efficiency can be substantially enhanced by exploiting its native matrix multiplication capability—a feature absent in prior FHE schemes. Conventional CtS and StC algorithms rely on multiple inter-slot operations performed via rotations. This approach incurs numerous key-switching steps, which constitute the primary runtime bottleneck. Leveraging the linearity of these transformations, we consolidate the CtS and StC operations into a single matrix multiplication, reducing the cost to only three key-switching operations.

1.1 Contributions

This work has three following contributions.

– **Generalizing the GL scheme beyond power-of-two dimensions.** We extend the GL scheme to support matrix arithmetic over non-power-of-two

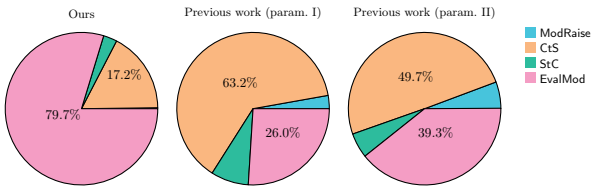

Fig. 1. Bootstrapping cost breakdown: CtS/StC accounts for a much smaller fraction in our method than in prior CKKS bootstrapping; the CtS depth is 2 for ours and 2/4 for previous work (parameter sets I/II defined in Section 5).

quotient rings. Specifically, we define matrix multiplication via the trace operation within $\mathcal{R}$-algebra, where $\mathcal{R}$ is a commutative ring. We then derive explicit formulas for matrices of dimension $\varphi(p) \times \varphi(p)$, where $p = r^k$ for a prime r and φ is the Euler totient function.

- **Bootstrapping tailored to the GL scheme.** We implement the roles of the classical CtS and StC procedures using the GL scheme's native matrix-multiplication primitive, requiring only a constant number of key-switching operations. The resulting amortized key-switching complexity is $O(\log(n))$, where n is the total number of slots. This improves upon the existing CtS and StC method, which has complexity $O(\delta \cdot n^{1/2\delta} \log n)$ where δ is typically chosen in $\{2, 3, 4\}$.
- **Practical optimizations for GL bootstrapping.** We propose an optimized variant of the GL bootstrapping that reduces the number of key switching operations from four to three. In addition, we remove the additional key switching introduced by sparse-secret encapsulation [10], thereby further reducing the computational overhead.

Fig. 1 shows that our bootstrapping significantly reduces the CtS/StC overhead. The key insight is that, among the four classical steps, CtS and StC can be implemented using the GL scheme's native matrix-multiplication primitive. The remaining challenge is homomorphic modular reduction in an *inside-out* manner, which remains an independent research interest.

1.2 Technical Overview

The GL scheme uses the plaintext ring $R' = \mathbb{Z}[i][X, Y, W]/\langle X^n - i,\, Y^n - i,\, \Phi_p(W)\rangle$, where Φ_p is the p-th cyclotomic polynomial, n is a power of two, and p is odd. For simplicity, we assume p is a power of an odd prime so that $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic with generator γ. Throughout this overview, we work over $\mathbb{C}$ and view all roots of unity as complex numbers.

Slots and canonical embedding. Let ζ be a primitive $4n$-th root of unity and set $\zeta_j := \zeta^{5^j}$ for $j \in [0, n)$. Let η be a primitive p-th root of unity; we enumerate primitive p-th roots as $\eta_\ell := \eta^{\gamma^\ell}$ for $\ell \in [0, \varphi(p))$. A plaintext polynomial $m \in R'$

encodes $\varphi(p)$ matrices of size $n \times n$ via the canonical embedding [27], i.e.,

$$m(\zeta_j, \zeta_k, \eta_\ell) = M^{(\ell)}_{j,k}, \qquad j,k \in [0,n),\ \ell \in [0,\varphi(p)).$$

Intuitively, X and Y index row/column coordinates and W indexes the batched dimension. We refer to the canonical embedding $R' \to \mathbb{C}^{n\times n\times\varphi(p)}$, which maps coefficients to *slots* (up to a scaling factor to preserve precision), as *decoding*; it is a ring homomorphism. The *encoding* is the inverse process.

Trace for matrix multiplication. Homomorphic addition and multiplication are the backbone of SIMD homomorphic encryption. Beyond these basic ring operations, GL (and our generalization) exploits an additional homomorphic primitive obtained by lifting plaintexts to a suitable algebra, multiplying there, and contracting back via a trace map. Let $\mathcal{R}$ be a commutative ring and let $\mathcal{S} = \mathcal{R}[Z]/\langle f(Z)\rangle$ be a finite $\mathcal{R}$-algebra with automorphism group G fixing $\mathcal{R}$. The trace map $\mathrm{Tr}_Z : \mathcal{S} \to \mathcal{R}$ is defined by $\mathrm{Tr}_Z(a) = \sum_{\sigma\in G} \sigma(a)$. For the homomorphisms underlying our slot semantics, applying Tr_Z at the encoded polynomial corresponds to summing over the appropriate slots.

This turns Tr_Z into a convenient contraction primitive. Given two plaintexts $u, v \in \mathcal{R}$, we embed them into $\mathcal{S}$ as $u', v^* \in \mathcal{S}$ so that Z plays the role of the summation index. Here, u' and v^* denote fixed involutions or rewiring chosen to align indices (e.g., the one induced by swapping X and Y, which corresponds to matrix transposition at the slot level). With these embeddings, the contraction $\mathrm{Tr}_Z(u' \cdot v^*)$ realizes matrix multiplication on slots, matching $\sum_t U^{(\ell)}_{j,t} V^{(\ell)}_{t,k}$ in each slot (j,k,ℓ), where $U^{(\ell)}$ and $V^{(\ell)}$ are the matrices encoded by u and v. Crucially, since Tr_Z is $\mathcal{R}$-linear and decryption consists of addition and multiplication over $\mathcal{R}$, *we can evaluate* Tr_Z *homomorphically.* Moreover, the specific bilinear form $(u, v) \mapsto \mathrm{Tr}_Z(u' \cdot v^*)$ that we construct admits an explicit implementation as plain matrix multiplication on the structured ciphertext coefficients over $\mathbb{Z}_q$.

We show this trace-based contraction for both $f(Z) = Z^n - i$ and $f(Z) = \Phi_p(Z)$. Since the Φ_p component contributes a factor of dimension $\varphi(p)$, which is not necessarily a power of two, this extends GL scheme's slotwise linear algebra beyond power-of-two cyclotomic components by enabling $\varphi(p)$ dimensions.

Bootstrapping and axis alignment. A GL ciphertext is an RLWE pair over $R'_q = R'/qR'$, i.e., $\mathsf{ct} \in (R'_q)^2$. Since the key is independent of Y, we can split $\mathsf{ct} = \sum_{j=0}^{n-1} \mathsf{ct}_j Y^j$ with $\mathsf{ct}_j \in (\mathbb{Z}[i][X,W]/\langle X^n - i, \Phi_p(W)\rangle)^2$. Bootstrapping ct then reduces to bootstrapping the n component ciphertexts ct_j. The CtS (resp. StC) transformation is realized by multiplying by an inverse DFT (resp. DFT) matrix along the slot axes. However, ciphertext–plaintext (ct–pt) matrix multiplication is natively expressed as right multiplication along the Y axis, whereas the required (I)DFT-style linear maps act along the X and W axes.

To implement these linear maps using right multiplication, we use a transpose trick. We introduce an auxiliary variable V with modulus $\Phi_p(V)$; it plays for the W-dimension the same structural role that Y plays for the X-dimension, allowing us to gather ciphertext components along an extra axis and realize

the non-power-of-two ($\varphi(p)$-length) transforms via right multiplication. We then apply a variable-swap automorphism ($X \leftrightarrow Y$ and $W \leftrightarrow V$ at the coefficient level) to transpose the relevant axes; at the slot level this corresponds to matrix transposition. This enables a *transpose* $\rightarrow$ *right multiplication* $\rightarrow$ *transpose back* sequence to apply the desired transform, and our method uses only a constant number of key-switching operations.

1.3 Related Works

Matrix multiplication has long been an important research topic in FHE [30, 32, 2, 40]. Prior work reduces homomorphic matrix multiplication to a few plaintext matrix multiplications: ct–pt in [37, 6], ciphertext–ciphertext (ct–ct) in Park's method [41], and its generalization via "slots-in-coefficient" encoding by Cheon et al. [15]. However, these approaches do not provide a fully slot-domain matrix multiplication or flexible matrix dimensions. In contrast, the GL scheme [27] leverages a multivariate ring and a Gaussian-integer viewpoint to enable a natural homomorphic matrix multiplication directly on slot values.

Following the initial work of Cheon et al. [14], CKKS bootstrapping has been studied extensively and has seen numerous improvements. The CKKS bootstrapping procedure follows the blueprint of Gentry-Halevi-Smart (GHS) [24] but performs a homomorphic evaluation of the modular reduction function, rather than bit extraction. After several levels of multiplication and rescaling, the ciphertext modulus is depleted to q_0. The goal of bootstrapping is to refresh this ciphertext, producing a new ciphertext with a larger modulus $Q \gg q_0$ that encrypts (approximately) the same plaintext m.

The process begins by lifting the modulus of ct to Q. However, the underlying plaintext then contains a large multiple of the modulus, represented as $m + q_0 I$. Bossuat et al. [10] proposed sparse-secret encapsulation to reduce $\|I\|_\infty$.

Subsequently, the term $q_0 I$ must be removed homomorphically. The CtS operation is performed—essentially a homomorphic inverse DFT—to obtain an encryption where the slots contain the coefficients of the polynomial $m + q_0 I$. Originally, $\tilde{O}(\sqrt{n})$ rotations are required for this homomorphic linear transformation via matrix multiplication [14]. Later, Chen et al. [13] introduced an FFT-like algorithm that offers a depth-runtime trade-off; by consuming a depth of δ, the runtime is reduced to $\tilde{O}(\delta n^{1/2\delta})$. Further, the *double hoisting* technique halves the cost of each FFT stage by rearranging the rotation sub-procedures [9].

The next step evaluates a periodic function that approximates reduction modulo q_0, often via the scaled sine $\frac{q_0}{2\pi} \sin\left(\frac{2\pi}{q_0} x\right)$ or its variants. Cheon et al. [14] used a Taylor series of the exponential function and extracted the imaginary part; Chen et al. [13] later proposed a Chebyshev approximation for improved efficiency and numerical stability. Han and Ki [31] used double-angle formulas for faster cosine evaluation, and subsequent works proposed low-degree compositions with arcsine [34], direct approximation methods [35], and iterative residual-noise bootstrapping to refine precision [5]. Then bootstrapping is completed with StC, the inverse of CtS.

A concept related to our method, known as batch bootstrapping, was proposed by Bae et al. [6]. They developed efficient ct–pt matrix multiplication for CKKS and used it to bootstrap multiple ciphertexts simultaneously. In the GL scheme [27], a single ciphertext can be viewed as n independent ciphertexts over a smaller ring, so bootstrapping is inherently batched. Using the *powerful basis* representation [39], one can perform CtS and StC with respect to each variable. Unlike Bae et al., which requires a batch of $O(N)$ ciphertexts for maximal efficiency, our method operates on a single GL ciphertext (corresponding to $O(\sqrt{N})$ "small" ciphertexts), thereby reducing the matrix size and significantly lowering the plaintext matrix-multiplication overhead.

1.4 Organization

The rest of the paper is organized as follows. Section 2 provides preliminaries, including an overview of the GL scheme and classical bootstrapping. Section 3 generalizes the GL scheme to non-power-of-two matrix arithmetic via a trace over commutative rings. In Section 4, we propose an efficient bootstrapping procedure with $O(1)$ key-switching operations for the coefficient–slot transformations. Section 5 reports implementation results and further optimization techniques, and Section 6 concludes with remarks on future work.

2 Preliminaries

2.1 Basic Notation

The ring of Gaussian integers is $\mathbb{Z}[i] = \{a+bi \mid a, b \in \mathbb{Z}\}$, canonically isomorphic to $\mathbb{Z}[I]/\left\langle I^2+1\right\rangle$. For $a \in \mathbb{Z}[i]$ or $\mathbb{C}$, let $\overline{a}$ denote complex conjugation, extended component-wise to polynomials and vectors. Let $q \in \mathbb{Z}_{>0}$ and $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$. For $a \in \mathbb{Z}_q$, let $[\![a]\!]$ be the unique representative in $(a + q\mathbb{Z}) \cap [-q/2, q/2)$, extended component-wise to polynomials and vectors; when unambiguous, we identify a with $[\![a]\!]$. We write $\|\cdot\|_2$ and $\|\cdot\|_\infty$ for the ℓ_2 and ℓ_∞ norms, respectively; for a polynomial, the norm is taken over its coefficient vector.

Let n be a power of two and p an odd prime power. Define γ as the generator for the cyclic group $\mathbb{Z}_p^*$. We define the polynomial ring

$$R := \mathbb{Z}[i][X, W]/\left\langle X^n - i,\, \Phi_p(W)\right\rangle \cong \mathbb{Z}[I, X, W]/\left\langle I^2+1,\, X^n - I,\, \Phi_p(W)\right\rangle,$$

We denote $R_q = R/qR$, and throughout the paper, we identify any element of R_q with its representative in R with coefficients in $[-q/2, q/2)$. The coefficient extract operation $[Z^i]h$ denotes the coefficient of Z^i in the polynomial h.

Throughout, for a multivariate polynomial f, we write $f(U, V, \dots)$ to denote the polynomial obtained from $f(X, Y, \dots)$ by the variable substitution $(X, Y, \dots) \mapsto (U, V, \dots)$. In particular, when working in quotient rings, this denotes the image under the induced R-algebra homomorphism determined by the same substitution.

For a ring $\mathcal{R}$ and a positive integer n, we define the matrix space $\mathcal{M}_n(\mathcal{R}) := \{(a_{jk}) \mid a_{jk} \in \mathcal{R}, 0 \le j,k < n\}$, the set of all $n \times n$ matrices over $\mathcal{R}$. Matrices are denoted in uppercase, e.g., M. The conjugate transpose (Hermitian transpose) of a matrix $M \in \mathcal{M}_n(\mathbb{C})$ or $\mathcal{M}_n(\mathbb{Z}_q[i])$ is denoted by M^{H}. The (j,k)-th entry of a matrix M is denoted by M_{jk} or $[M]_{jk}$. We define a matrix with its elements as $M = [m(i,j)]_{i,j\in[0,n)}$ where its (i,j)-th entry is $m(i,j)$. Similarly, we define a vector with its elements as $v = (v_j)_{j\in[0,n)}$, and we may ignore the indices in the bottom when it is obvious from the context.

2.2 The Gentry-Lee Scheme

Encoding Structure and Matrix Multiplication While the GL scheme supports integer matrices with BGV/BFV-like encoding, this work focuses on CKKS-like encoding version for approximate arithmetic. The plaintext space of the GL scheme is $R' = R[Y]/\langle Y^n - i\rangle \cong \mathbb{Z}[i][X,Y,W]/\langle X^n - i, Y^n - i, \Phi_p(W)\rangle$. p can be arbitrary odd number, but we focus on the case where $\mathbb{Z}_p^*$ has a single generator γ for simplicity. We use the canonical embedding for decoding $\tau : R' \to \mathcal{M}_n(\mathbb{C})^{\varphi(p)}$, $m \mapsto (M^{(0)}, \ldots, M^{(\varphi(p)-1)})$, where each $M^{(\ell)} \in \mathcal{M}_n(\mathbb{C})$ is defined as:

$$m(\zeta_j, \zeta_k, \eta_\ell) = M^{(\ell)}_{jk}, \quad j,k \in [0,n),\ \ell \in [0,\varphi(p)),$$

where $M^{(\ell)}_{jk}$ denotes the (j,k)-th entry of the ℓ-th matrix $M^{(\ell)}$. Here, $\zeta_j = \zeta^{5^j}$ and $\eta_\ell = \eta^{\gamma^\ell}$, where ζ and η are primitive $4n$-th and p-th roots of unity, respectively. The encoding is the inverse map of τ followed by rounding to R'; however, precision is lost due to this rounding. To solve this problem, we apply the scaling factor technique from the CKKS scheme. For a scaling factor Δ, we define the encoding and decoding as follows:

$$\mathsf{encode}((M^{(\ell)})_\ell; \Delta) := \left\lfloor \Delta \cdot \tau^{-1}\big((M^{(\ell)})_\ell\big) \right\rceil_{R'_q} \text{ and } \mathsf{decode}(m; \Delta) := \frac{1}{\Delta} \cdot \tau(m).$$

Remark 1. The canonical embedding is a variant of the Discrete Fourier Transform (DFT); the encoding and decoding procedures are equivalent to the inverse DFT and DFT, respectively. We often use FFT algorithms to speed up the encoding and decoding, or equivalently, the matrix multiplication operation.

Then, by defining the trace operation $Tr_Z(a) := [Z^0]a$, we have to following matrix multiplication operation.

Theorem 1 (Batched matrix multiplication [27]). *Let* $c = Tr_Z\big(a(X,Z,W) \cdot \overline{b}(Y^{-1}, Z^{-1}, W^{-1})\big)$, *where* $a(\zeta_j,\zeta_k,\eta_\ell) = A^{(\ell)}_{jk}$ *and* $b(\zeta_j,\zeta_k,\eta_\ell) = B^{(\ell)}_{jk}$. *Then*

$$c(\zeta_j,\zeta_k,\eta_\ell) = \frac{1}{n}\big[A^{(\ell)} B^{(\ell)\mathrm{H}}\big]_{jk}.$$

Encryption and Security We define a ciphertext $\mathsf{ct}_s(m) = (b, a) \in (R'_q)^2$ under the secret key $s \in R$ such that it satisfies $b + a \cdot s = m + e \in R'_q$. Here, $s \leftarrow \chi_{key}$, $e \leftarrow \chi_{err}$, and a is uniformly sampled from R'_q; χ_{key} and χ_{err} denote distributions over R and R', respectively. Since the key is sampled from R, we can view the ciphertext as n independent RLWE ciphertexts over R, namely [27]:

$$\mathsf{ct}_s^{(j)}(m^{(j)}) = \big([Y^j]b, [Y^j]a\big) \in R_q^2.$$

Consequently, the security relies on the RLWE assumption over $\mathbb{Z}[X]/\langle \Phi_{4np}(X)\rangle$; we refer the reader to [38, 27] for further details.

Encrypted Matrix Multiplication Let $m, \mu \in R'$ be two plaintexts encoding different matrices. Consider a ciphertext $\mathsf{ct}_s(m) = (b, a) \in R'^2_q$; we define the ct–pt matrix multiplication as:

$$\mathsf{ct}_s(m \circledast \mu) = (b \circledast \mu, a \circledast \mu),$$

where $u \circledast v := \mathrm{Tr}_Z\left(u(X, Z, W) \cdot \overline{v}(Y^{-1}, Z^{-1}, W^{-1})\right)$. The key observation in the GL scheme is that the trace operation is R-linear. Since $s \in R$, we can verify the decryption as follows:

$$(b \circledast \mu) + (a \circledast \mu) \cdot s = (b + a \cdot s) \circledast \mu = (m + e) \circledast \mu.$$

Given the second ciphertext $\mathsf{ct}_s(\mu) = (\alpha, \beta)$, the ct–ct multiplication also follows naturally:

$$(b \circledast \beta, a \circledast \beta, b \circledast \alpha, a \circledast \alpha).$$

Using the transposed secret key $s^* = \overline{s}(X^{-1}, Y^{-1}, W^{-1})$ for decryption, we have:

$$\begin{aligned}&(b \circledast \beta) + (a \circledast \beta) \cdot s + (b \circledast \alpha) \cdot s^* + (a \circledast \alpha) \cdot s \cdot s^* \\ &= (b + a \cdot s) \circledast (\beta + \alpha \cdot s^*) = (m + e) \circledast (\mu + e').\end{aligned}$$

Addition and Hadamard Product Encrypted addition and the Hadamard product follow naturally from the fact that the canonical encoding $R' \to \mathbb{C}^{n \times n \times \varphi(p)}$ is a ring homomorphism. Hence, $m+\mu$ and $m \cdot \mu$ in R' correspond to the addition and Hadamard product of their encoded matrices, respectively. The addition of two ciphertexts $\mathsf{ct}_s(m) = (b, a)$ and $\mathsf{ct}_s(\mu) = (\beta, \alpha)$ is defined as:

$$\mathsf{ct}_s(m + \mu) = (b + \beta, a + \alpha).$$

The Hadamard product of two ciphertexts is defined as:

$$\mathsf{ct}_s(m \cdot \mu) = (b \cdot \beta, a \cdot \beta + b \cdot \alpha, a \cdot \alpha).$$

Its decryption is verified as follows:

$$(b \cdot \beta) + (a \cdot \beta + b \cdot \alpha) \cdot s + (a \cdot \alpha) \cdot s^2 = (b + a \cdot s) \cdot (\beta + \alpha \cdot s) = (m + e) \cdot (\mu + e').$$

Rotation and Transposition Rotation and matrix transposition are important and practically useful features of the GL scheme. The Galois automorphism [25] enables both inter- and intra-matrix rotation. To rotate rows by an amount δ, we substitute X with X^{5^δ}. Under the decoding map $m(X, Y, W) \mapsto \big([m(\zeta_j, \zeta_k, \eta_\ell)]_{j,k\in[0,n)}\big)_{\ell\in[0,\varphi(p))}$, this substitution yields (indices are taken modulo n):

$$\big(\big[m(\zeta_j^{5^\delta}, \zeta_k, \eta_\ell)\big]_{j,k}\big)_\ell = \big(\big[m(\zeta_{j+\delta}, \zeta_k, \eta_\ell)\big]_{j,k}\big)_\ell,$$

since $\zeta_j = \zeta^{5^j}$. Similarly, we can rotate columns by substituting Y with Y^{5^δ}, and rotate batched matrices (inter-matrix rotation) by substituting W^{γ^δ}, where γ is a generator of $\mathbb{Z}_p^*$. As those substitutions are automorphism maps, we use the isomorphism to preserve the operation for ciphertexts. In other words, given $\mathsf{ct}_s(m) = (b, a)$, where $m + e = b + a \cdot s$, we have the rotated ciphertext $(b(X^{5^\delta}, Y, W), a(X^{5^\delta}, Y, W))$, which is decrypted as:

$$b(X^{5^\delta}, Y, W) + a(X^{5^\delta}, Y, W) \cdot s(X^{5^\delta}, W) = (m + e)(X^{5^\delta}, Y, W).$$

Similarly for Y and W, by substuting Y with Y^{5^δ} and W with W^{γ^δ}. It is worth noting that the key remains the same for the Y case as the secret key is a constant with respect to Y.

Matrix transposition is performed similarly: we substitute (X, Y) with (Y, X) for transposition, and (i, X, Y, W) with $(-i,\, Y^{-1},\, X^{-1},\, W^{-1})$ for Hermitian transpose. Accordingly, the secret key becomes $s(Y, W)$ and $s^* = \overline{s}(X^{-1}, Y^{-1}, W^{-1})$ for transposition and Hermitian transposition, respectively. For a polynomial $a \in R'$, we denote these operations by $\mathsf{trans}(a)$ and $\mathsf{Herm}(a)$, respectively. Given a ciphertext $\mathsf{ct}_s(m) = (b, a)$ satisfying $m+e = b+a\cdot s$, the transposed ciphertext is $(\mathsf{trans}(b), \mathsf{trans}(a))$, which decrypts as:

$$\mathsf{trans}(b) + \mathsf{trans}(a) \cdot s(Y, W) = \mathsf{trans}(m + e).$$

Similarly, the Hermitian transposed ciphertext is decrypted as:

$$\mathsf{Herm}(b) + \mathsf{Herm}(a) \cdot s^* = \mathsf{Herm}(m + e).$$

Key Switching The form of the decryption key s evolves depending on the homomorphic operation applied. Specifically, linear operations induce automorphisms on the secret key: Hermitian transpose yields s^*, while standard transpose results in $s(Y, W)$. Similarly, row and inter-matrix rotations transform the key into $s(X^{5^k}, W)$ and $s(X, W^{\gamma^k})$, respectively. On the other hand, multiplicative operations increase the key degree or dimension: Hadamard product results in (s, s^2), whereas matrix multiplication expands the key to (s, s^*, ss^*). Consider a ciphertext (d_0, d_1, d_2) encrypting a message m under the secret key (s, s^2). If we can obtain a ciphertext encrypting $d_2 \cdot s^2$, we can find the ciphertext of m under secret s as:

$$(d_0, d_1) + \mathsf{ct}_s(d_2 \cdot s^2) = \mathsf{ct}_s(d_0 + d_1 \cdot s) + \mathsf{ct}_s(d_2 \cdot s^2) = \mathsf{ct}_s(d_0 + d_1 \cdot s + d_2 \cdot s^2).$$

The encryption of s^2 is provided by the key owner; similarly, the encryption of other keys can be obtained. Let s' denote a secret key we switch from.

The main challenge in key switching lies in computing $\mathsf{ct}_s(a \cdot s')$ with *small noise*, where a is a element of ciphertext such as d_2 above. We use the *decomposition* technique and RLWE' ciphertexts for the key-switching process [28, 31]. Assume there exists a decomposition vector $\boldsymbol{g} = \left(g_0, \ldots, g_{d_g-1}\right) \in \mathbb{Z}^{d_g}$ and a decomposition function $h : \mathbb{Z} \to \mathbb{Z}^{d_g}$ satisfying the following:

1. For $a \in R$, $\|h(a)\| \leq B$ for some bound B.
2. $\langle h(a), \boldsymbol{g} \rangle = a$, or at least $\langle h(a), \boldsymbol{g} \rangle \approx a$ with negligible noise.

Note that this extends naturally to polynomials coefficient-wise. For details on various decomposition techniques, we refer to [12, 20, 19, 18, 36].

The RLWE' ciphertexts are defined as a set of ciphertexts:

$$\mathrm{RLWE}'_s(s') = \left(\mathsf{ct}(g_0 s'), \ldots, \mathsf{ct}(g_{d_g-1} s')\right).$$

Then we define the outer product $\odot : R \times \mathrm{RLWE}' \to \mathrm{RLWE}$:

$$a \odot \mathrm{RLWE}'(s') = \left\langle h(a), \mathrm{RLWE}'(s') \right\rangle = \mathsf{ct}\left(\textstyle\sum_i a_i g_i s'\right) = \mathsf{ct}(as').$$

Due to the first characteristic of decomposition (small norm), the noise introduced by the outer product remains small; however, it is still much larger than that of a fresh ciphertext. Therefore, we temporarily increase the modulus to qP for some integer P (source secret is pre-scaled by P within RLWE'), and scale down the ciphertext after the outer product, a technique known as modulus switching [26, 16]. We *define the process* $\mathsf{Switch}(a, \mathsf{ksk}_{s' \mapsto s})$ to output $\mathsf{ct}_s(as') \in R_q^2$.

The GL scheme uses two different key-switching functions depending on the source secret key: $\mathsf{Switch}_{\mathsf{small}} : R' \times R^{2d_g} \mapsto R'^2$ and $\mathsf{Switch}_{\mathsf{big}} : R' \times R'^{2d_g} \mapsto R'^2$. The key switching $\mathsf{Switch}_{\mathsf{small}}$ is used when the source secret key is in R, whereas $\mathsf{Switch}_{\mathsf{big}}$ is used when the source secret key is in the extended ring R'. The destination secret key is always in R.

Given a polynomial $a \in R'$ and a key-switching key from $s'(X, W) \in R$ to $s(X, W) \in R$, $\mathsf{ksk}_{s'(X,W) \mapsto s(X,W)} := \mathrm{RLWE}'_{s(X,W)}(Ps'(X, W)) \in R_{qP}^{2d_g}$, the operation $\mathsf{Switch}_{\mathsf{small}}$ is defined as:

$$\mathsf{Switch}_{\mathsf{small}}(a, \mathsf{ksk}_{s'(X,W) \mapsto s(X,W)}) = \textstyle\sum_{j=0}^{n-1} \mathsf{Switch}\left(a_j, \mathsf{ksk}_{s'(X,W) \mapsto s(X,W)}\right) \cdot Y^j,$$

where $a(X, Y, W) = \sum_{j=0}^{n-1} a_j(X, W) Y^j$. One may regard $\mathsf{ksk}_{s'(X,W) \mapsto s(X,W)}$ as the key-switching key for R' with zero Y-degree.

Similarly, given a polynomial $a \in R'$ and a key-switching key from $s'(X, Y, W)$ $\in R'$ to $s(X, W) \in R$, where

$$\mathsf{ksk}_{s'(X,Y,W) \mapsto s(X,W)} := \mathrm{RLWE}'_{s(X,W)}(Ps'(X, Y, W)) \in R_{qP}^{\prime 2d_g},$$

the operation $\mathsf{Switch}_{\mathsf{big}}$ is defined as:

$$\mathsf{Switch}_{\mathsf{big}}(a, \mathsf{ksk}_{s'(X,Y,W) \mapsto s(X,W)}) = \mathsf{Switch}\left(a, \mathsf{ksk}_{s'(X,Y,W) \mapsto s(X,W)}\right).$$

In the GL scheme, we use $\mathsf{Switch}_{\mathsf{small}}$ for rotations and Hadamard product, and $\mathsf{Switch}_{\mathsf{big}}$ for matrix multiplication and transposition.

Given two key-switching keys $\mathsf{ksk}_{s(Y,W)\mapsto s(X,W)}$ and $\mathsf{ksk}_{s^*\mapsto s}$, we denote the transposition and Hermitian transposition operations for a ciphertext $\mathsf{ct} \in R'^2$ as $\mathsf{trans}(\mathsf{ct}, \mathsf{ksk}_{s(Y,W)\mapsto s(X,W)})$ and $\mathsf{Herm}(\mathsf{ct}, \mathsf{ksk}_{s^*\mapsto s})$, respectively.

Rescaling We realize that the scaling factor is also multiplied with the message; the problem is that the scaling factor grows exponentially, $\Delta \to \Delta^2 \to \Delta^4 \to \cdots$. Hence, in the CKKS scheme, a novel *rescaling* technique is used to control this growth linearly [16], and GL also adopts this technique. Let $q = \prod_j q_j$. Then for a $q_j | q$, we define the rescaling as follows:

$$\mathsf{rescale}(\cdot) : \mathbb{Z}_q \to \mathbb{Z}_{q/q_j}, \quad a \mapsto \lfloor a/q_j \rceil \pmod{q/q_j},$$

extended to polynomials and ciphertexts coefficient-wise. By choosing $q_j \approx \Delta$, we can control the noise growth as

$$\Delta \xrightarrow{\text{mult}} \Delta^2 \xrightarrow{\mathsf{rescale}} \Delta^2/q_L \approx \Delta \xrightarrow{\text{mult}} \Delta^2 \xrightarrow{\mathsf{rescale}} \Delta^2/q_{L-1}.$$

We note that rescaling decreases the ciphertext modulus; hence, we can perform multiplications only by a certain amount, the so-called *level* of a ciphertext. When the level reaches zero, we need to increase the ciphertext modulus for further computation, which we call *bootstrapping*.

2.3 Bootstrapping for the CKKS Scheme

Here we briefly explain the standard bootstrapping procedure based on transformations between slot and coefficient representations [24]. Specifically, we focus on its CKKS instantiation [14] because it is efficient and practically useful. It is composed of four steps as follows:

- ModRaise: The plaintext polynomial $m \pmod{q_0}$ becomes $m' = m + q_0 I \pmod{Q}$ for a sufficiently large Q, where I is a random integer introduced by ignoring the modular reduction. The following procedure removes this quotient term $q_0 I$ while preserving a certain amount of modulus, reduced from Q, so that we can perform further operations.
- CtS: While our goal is to apply the reduction procedure of $q_0 I$, namely EvalMod, coefficient-wise, we can only perform Hadamard product on slots. Hence, we perform homomorphic *encoding* so that the slots hold the coefficients of m'.
- EvalMod: Set the scaling factor to q_0, then perform the mod 1 operation using approximate polynomial evaluation (thus only addition and multiplication) of a proper periodic function on the slots of both ciphertexts.
- StC: In this step, we perform homomorphic *decoding* on the slots.

Noise and Complexity Bottleneck While EvalMod is the noise and level bottleneck, the CtS and StC steps are usually the runtime bottleneck, as they require many rotation operations. The large-radix FFT-like method [13, 9] is widely used, providing a level-runtime trade-off. We usually set parameters so that CtS and StC have a comparable runtime to EvalMod by using more levels.

3 Generalization of Gentry-Lee Scheme

To establish a rigorous foundation for our proposed bootstrapping and the generalization of the GL method, we begin by formally defining the trace operation. Starting from the classical definition over commutative rings, we show that the essential properties for our construction follow elegantly without heuristic assumptions.

3.1 The Algebraic Trace

In this paper, we make use of a trace map on certain quotient R-algebras. We begin with a general notion of the trace map obtained by summing over a finite subgroup of R-algebra automorphisms, cf. Greither §0 [29].

Definition 1 (Trace map via automorphisms). *Let $\mathcal{R}$ be a commutative ring and let $\mathcal{S}$ be an $\mathcal{R}$-algebra.*[4] *Let $\mathrm{Aut}_{\mathcal{R}}(\mathcal{S})$ denote the group of all $\mathcal{R}$-algebra automorphisms of $\mathcal{S}$ (i.e., ring automorphisms fixing $\mathcal{R}$ pointwise). Given a finite subgroup $G \subset \mathrm{Aut}_{\mathcal{R}}(\mathcal{S})$, we define the G-trace by*

$$\mathrm{Tr}^{G}_{\mathcal{S}/\mathcal{R}}(x) := \textstyle\sum_{\sigma\in G} \sigma(x), \qquad x \in \mathcal{S}. \tag{1}$$

This map produces values in the fixed subalgebra $\mathcal{S}^G$ (i.e., $\mathrm{Tr}^{G}_{\mathcal{S}/\mathcal{R}}(x) \in \mathcal{S}^G$ for all $x \in \mathcal{S}$), where S^G is defined as $\{s \in S \mid \sigma(s) = s \text{ for all } \sigma \in G\}$.

Remark 2. $\mathcal{S}/\mathcal{R}$ is a called G-Galois ring extension only if $\mathcal{S}^G = \mathcal{R}$.

Remark 3 (Generalization of the Galois Field trace). Our definition generalizes the trace over field. For instance, when $\mathcal{S}/\mathcal{R}$ is a finite field extension $\mathbb{F}_{q^n}/\mathbb{F}_q$, G is generated by the Frobenius automorphism $x \mapsto x^q$, and $\mathrm{Tr}(x) = \sum_{i=0}^{n-1} x^{q^i}$.

Based on this general framework, we now specialize the definition to the quotient algebra $\mathcal{S} = \mathcal{R}[Z]/\langle f(Z)\rangle$. Specifically, we utilize the subgroup of *power automorphisms* to define the specific trace map used in our scheme.

Definition 2 (Power-trace on a polynomial quotient algebra). *Let $\mathcal{R}$ be a commutative ring and let $f(Z) \in \mathcal{R}[Z]$ be a monic polynomial of degree d. Set*

$$\mathcal{S} := \mathcal{R}[Z]/\langle f(Z)\rangle,$$

[4] $\mathcal{S}$ is a commutative ring equipped with a ring homomorphism $\phi : \mathcal{R} \to \mathcal{S}$, which induces scalar multiplication $r \cdot s := \phi(r)s$ on $\mathcal{S}$ for $r \in \mathcal{R}$.

where we denote the residue class of Z simply by Z. Define the subgroup of power automorphisms *by*

$$G_{\mathrm{pow}} := \{\sigma \in \mathrm{Aut}_{\mathcal{R}}(\mathcal{S}) \mid \sigma(Z) = Z^k, k \in \mathbb{Z} \setminus \{0\}\}.$$

We define the Z-trace *(or* power-trace*) by*

$$\mathrm{Tr}_Z(a) := \mathrm{Tr}_{\mathcal{S}/\mathcal{R}}^{G_{\mathrm{pow}}}(a) = \textstyle\sum_{\sigma \in G_{\mathrm{pow}}} \sigma(a), \qquad a \in \mathcal{S}.$$

Or equivalently, for the power index set

$$I_{\mathrm{pow}} := \{k \in \mathbb{Z} \mid \sigma : Z \mapsto Z^k \in G_{\mathrm{pow}}\}, \quad \textit{we have} \quad \mathrm{Tr}_Z(a) = \textstyle\sum_{k \in I_{\mathrm{pow}}} a(Z^k).$$

Here the index set I_{pow} parametrizes distinct power automorphisms; different integers k giving rise to the same automorphism are identified implicitly.

Remark 4. In general, the trace defined in Definition 1 outputs values in the fixed subalgebra $\mathcal{S}^G$. However, in our specific construction of $f(Z)$, we need the stronger property that the power-trace Tr_Z is in fact $\mathcal{R}$-valued, i.e., $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$.

Remark 5. This definition generalizes the $\mathbb{Q}$-trace map used in [39] and retains its properties. However, we need such a generalization because we need a trace map over the ring $\mathcal{R} = \mathbb{Z}[X_1, \ldots, X_n]/\langle f_1(X_1), \ldots, f_n(X_n)\rangle$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ (with the corresponding reduction of $\mathcal{S}$).

Specifically, we use the following $\mathcal{R}$-algebras with useful properties for our bootstrapping construction. Proofs not included in the main text are deferred to the Appendix.

Lemma 1 (Power maps in the quotient by $Z^n - i$). *Let $\mathcal{R}$ be a commutative ring containing an element $i \in \mathcal{R}$ with $i^2 = -1$, and let n be a power of two such that $n \in \mathcal{R}^*$, i.e., n is invertible in $\mathcal{R}$. Set $\mathcal{S} := \mathcal{R}[Z]/\langle Z^n - i\rangle$. Then, the group of power automorphisms is isomorphic to the index set*

$$G_{\mathrm{pow}} \cong I_{\mathrm{pow}} := \{k \in \mathbb{Z}_{4n}^* \mid k \equiv 1 \pmod 4\},$$

and the trace map satisfies $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$. Specifically, $\mathrm{Tr}_Z(a) = n \cdot [Z^0]a$. Since $n \in \mathcal{R}^*$, the trace does not vanish on non-zero constants.

Remark 6. We note that the underlying ring structure is identical to that of GL scheme. The trace in [27], defined as the constant term a_0, coincides with our definition (normalized by the factor n).

Lemma 2 (Power maps in the cyclotomic quotient). *Let $p \in \mathcal{R}^*$ be an odd prime. Set $\mathcal{S} := \mathcal{R}[Z]/\langle \Phi_p(Z)\rangle$. Then, $G_{\mathrm{pow}} \cong \mathbb{Z}_p^*$, and*

$$\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}.$$

Specifically, $\mathrm{Tr}_Z(a) = p \cdot a_0 - \sum_{j=0}^{\varphi(p)-1} a_j$, where $a = \sum_{j=0}^{\varphi(p)-1} a_j Z^j$.

The above lemma is a special case of Lemma 3

Lemma 3 (Power maps in the prime-power cyclotomic quotient). *Let $\mathcal{R}$ be a commutative ring and let p be an odd prime with $p \in \mathcal{R}^*$. Fix $k \geq 1$ and set $\mathcal{S} := \mathcal{R}[Z]/\langle \Phi_{p^k}(Z) \rangle$. Then $G_{\mathrm{pow}} \cong \mathbb{Z}_{p^k}^*$ and $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$. More precisely, writing $a = \sum_{j=0}^{\varphi(p^k)-1} a_j Z^j$ (with $\deg < \varphi(p^k)$), we have*

$$\mathrm{Tr}_Z(a) = p^k \cdot a_0 - p^{k-1} \sum_{t=0}^{p-2} a_{tp^{k-1}} \in \mathcal{R}.$$

We rely on the following properties to relate the algebraic trace over $\mathcal{R}$ to the its evaluation over $\mathbb{C}$.

Lemma 4 (Trace Commutes with Homomorphisms). *Let $\psi : \mathcal{R} \to \mathcal{R}'$ be a ring homomorphism and let $f(Z) \in \mathcal{R}[Z]$ be monic. Set*

$$\mathcal{S} := \mathcal{R}[Z]/\langle f(Z) \rangle, \qquad \mathcal{S}' := \mathcal{R}'[Z]/\langle \psi(f)(Z) \rangle,$$

and let $\psi_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}'$ be the induced homomorphism on coefficients. Then, for all $a \in \mathcal{S}$,

$$\psi_{\mathcal{S}}\big(\mathrm{Tr}_Z(a)\big) = \mathrm{Tr}_Z\big(\psi_{\mathcal{S}}(a)\big),$$

where Tr_Z on $\mathcal{S}'$ is defined as in Definition 2. Moreover, if $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$, then

$$\psi\big(\mathrm{Tr}_Z(a)\big) = \mathrm{Tr}_Z\big(\psi_{\mathcal{S}}(a)\big).$$

Proof. It suffices to show that $\psi_{\mathcal{S}}$ commutes with every power automorphism. We refer to Section B for the detailed proof.

For the specific polynomials ($Z^n - i$ and $\Phi_p(Z)$) utilized in this paper, we can express the trace as a sum over roots as stated in Lemma 5.

Lemma 5 (Trace as a sum over roots). *Let $f \in \mathcal{R}[Z]$ be monic and set $S := \mathcal{R}[Z]/\langle f(Z) \rangle$. Assume that:*

1. *the power-trace is $\mathcal{R}$-valued, i.e., $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$;*
2. *f splits in $\mathcal{R}$ with distinct roots $\Omega = \{\zeta^k \mid k \in I_{\mathrm{pow}}\} \subset \mathcal{R}$, where $\zeta \in \Omega$ is fixed and $k \mapsto \zeta^k$ is a bijection $I_{\mathrm{pow}} \to \Omega$.*

Then, for any $a \in \mathcal{S}$, we have $\mathrm{Tr}_Z(a) = \sum_{\omega \in \Omega} a(\omega)$.

In this paper, we apply the result of Lemma 5 to the specific polynomials $f(Z) = Z^n - i$ and $f(Z) = \Phi_p(Z)$, for odd prime power p. For these cases, the set of roots can be explicitly described as $\{\zeta^k \mid k \in \mathbb{Z}_{4n}^*,\ k \equiv 1 \pmod 4\}$ (resp. $\{\eta^k \mid k \in \mathbb{Z}_p^*\}$), where ζ (resp. η) is a primitive $4n$-th (resp. p-th) root of unity. This structural property holds for any polynomial quotient ring whose coefficient ring is a subring of $\mathbb{C}$, and also for coefficient ring $\mathbb{Z}_q$ when $q \equiv 1 \pmod{4np}$. In particular, it covers multivariate quotients such as $\mathbb{Z}_q[X_1, \ldots, X_n, Z]/\langle g_1(X_1), \ldots, g_n(X_n), f(Z) \rangle$. Henceforth, we focus exclusively on these polynomials and their corresponding power-trace Tr_Z.

Lemma 4 also yields the following theorem, which connects the plaintext polynomials with integer coefficients to the ciphertext polynomials obtained by reducing coefficients modulo q (i.e., over $\mathbb{Z}_q$).

Corollary 1. *Let $[\cdot]_q : \mathbb{Z} \to \mathbb{Z}_q$ denote the natural quotient map modulo q, extended coefficient-wise to polynomials. Let $\mathcal{R} = \mathbb{Z}[X_1, \ldots, X_n]/\langle g_1(X_1), \ldots, g_n(X_n)\rangle$ and $\mathcal{S} = \mathcal{R}[Z]/\langle f(Z)\rangle$, where $f(Z)$ is either $Z^n - i$ or $\Phi_p(Z)$. Define the corresponding quotient rings as $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ and $\mathcal{S}_q = \mathcal{S}/q\mathcal{S}$.*

Then, for any $a \in \mathcal{S}$, the trace map commutes with modular reduction:

$$[\mathrm{Tr}_Z(a)]_q = \mathrm{Tr}_Z([a]_q).$$

Consequently, we can uniquely recover the trace from its modular image:

$$\mathrm{Tr}_Z(a) = [\![\mathrm{Tr}_Z([a]_q)]\!],$$

provided that the coefficients of $\mathrm{Tr}_Z(a)$ satisfy $\|\mathrm{Tr}_Z(a)\|_\infty < q/2$.

Proof. This follows directly from Lemma 4 by taking $\psi(\cdot) = [\cdot]_q$. □

3.2 Gentry-Lee Scheme Matrix Multiplication via Our Trace

In this subsection, to facilitate the understanding of our proposed method, we revisit the matrix multiplication of the GL scheme [27] using our trace definition. We recall the ring $R' = \mathbb{Z}[i][X, Y, W]/\langle X^n - i, Y^n - i, \Phi_p(W)\rangle$.

Theorem 2 (Revisiting Theorem 1 [27]). *Let $a, b \in R'$. We view them as elements in $R'[Z]/\langle Z^n - i\rangle$ by substituting $Y \mapsto Z$ and $(i, X, Y, W) \mapsto (-i, Y^{-1}, Z^{-1}, W^{-1})$ denoted by $a(X, Z, W)$ and $\overline{b}(Y^{-1}, Z^{-1}, W^{-1})$. Let $c = \mathrm{Tr}_Z\left(a(X, Z, W)\cdot\overline{b}(Y^{-1}, Z^{-1}, W^{-1})\right)$, where $a(\zeta_j, \zeta_k, \eta_\ell) = A^{(\ell)}_{jk}$ and $b(\zeta_j, \zeta_k, \eta_\ell) = B^{(\ell)}_{jk}$. Then*

$$c(\zeta_j, \zeta_k, \eta_\ell) = \left[A^{(\ell)} B^{(\ell)\mathrm{H}}\right]_{jk}.$$

Proof. For completeness, we reprove the theorem using our power-trace. Consider the partial decoding map $\Psi_{j,k,\ell} : R' \to \mathbb{C}$ (note $\Psi_{j,k,\ell}$ for all j, k, ℓ forms the decoding). Since it is a ring homomorphism, we may apply Lemma 4 to obtain

$$\begin{aligned}\Psi_{j,k,\ell}(c) &= \mathrm{Tr}_Z\left(\Psi_{j,k,\ell}\left(a(X, Z, W) \cdot \overline{b}(Y^{-1}, Z^{-1}, W^{-1})\right)\right)\\ &= \mathrm{Tr}_Z\left(a(\zeta_j, Z, \eta_\ell) \cdot \overline{b}(\zeta_k^{-1}, Z^{-1}, \eta_\ell^{-1})\right),\end{aligned}$$

where the Tr_Z on the right-hand side is defined over $\mathbb{C}[Z]/\langle Z^n - i\rangle$. By Lemma 5,

$$c(\zeta_j, \zeta_k, \eta_\ell) = \textstyle\sum_{\iota=0}^{n-1} a(\zeta_j, \zeta_\iota, \eta_\ell) \cdot \overline{b}(\zeta_k^{-1}, \zeta_\iota^{-1}, \eta_\ell^{-1}) \quad = \sum_{\iota=0}^{n-1} A^{(\ell)}_{j\iota} \cdot \overline{B^{(\ell)}_{k\iota}} = \left[A^{(\ell)} B^{(\ell)\mathrm{H}}\right]_{jk},$$

and the claim follows. □

We can also reprove the following theorem using the property that the trace can be calculated via matrix multiplication.

Theorem 3 (Trace by Matrix Multiplication [27]). *Let a, b be polynomials R', let us define the involution b^* as: $b^*(X,Y,W) := \overline{b}(X^{-1}, Y^{-1}, W^{-1})$. Define matrices $\hat{A}$ and $\hat{B'}$ in $\mathcal{M}_n(\mathbb{Z}[i][W]/\langle \Phi_p(W)\rangle)$ by extracting coefficients[5] corresponding to the canceling powers of Z:*

$$[\hat{A}]_{j\ell} = [X^j Y^\ell]a, \qquad [\hat{B'}]_{k\ell} = [X^k Y^{-\ell}]b^*.$$

Then, for $c(X,Y,W) = \mathrm{Tr}_Z\left(a(X,Z,W) \cdot b^*(Y,Z,W)\right)$, its coefficient corresponding to $X^j Y^k$ is given by:

$$[X^j Y^k]c = n \cdot [\hat{A}\hat{B'}^{\mathrm{T}}]_{jk}.$$

Proof. Let $d(X,Y,Z,W) = a(X,Z,W) \cdot b^*(Y,Z,W) \in R'_q[Z]/\langle Z^n - i\rangle$. Applying Lemma 5, we have:

$$\mathrm{Tr}_Z(d) = n \cdot [Z^0]d.$$

Let us denote the coefficients of a and b^* explicitly as:

$$a(X,Z,W) = \textstyle\sum_{j,\alpha\in[0,n)} a_{j,\alpha}(W) X^j Z^\alpha, \quad b^*(Y,Z,W) = \sum_{k,\beta\in[0,n)} b^*_{k,-\beta}(W) Y^k Z^{-\beta}.$$

The constant term $[Z^0]d$ arises from the product of terms where indices of Z sum to zero (i.e., $\beta = -\alpha$). Thus:

$$[Z^0]d = \textstyle\sum_{j,k} \sum_\alpha a_{j,\alpha} \cdot b^*_{k,-\alpha} \cdot X^j Y^k.$$

To match the matrix definitions where $[\hat{A}]_{j\alpha} = a_{j,\alpha}$ and $[\hat{B'}]_{k\alpha} = [Y^k Z^{-\alpha}]b^* = b^*_{k,-\alpha}$, we rewrite the sum for a fixed (j,k) term:

$$[X^j Y^k]([Z^0]d) = \textstyle\sum_{\alpha=0}^{n-1} [\hat{A}]_{j\alpha} \cdot [\hat{B'}]_{k\alpha}.$$

The right-hand side is precisely the (j,k)-entry of the matrix product $\hat{A}\hat{B'}^{\mathrm{T}}$. Multiplying by the scaling factor n, the result follows. □

Corollary 1 extends the above results to the modular setting over $R'_q = R'/qR'$, where the encrypted operations take place (i.e., $a, b \in R'_q$). Moreover, when a and b^* are represented in W-axis NTT form, the matrix product in $\mathcal{M}_n(\mathbb{Z}_q[i][W]/\langle \Phi_p(W)\rangle)$ can be computed using $\varphi(p)$ matrix multiplications over $\mathbb{Z}_q[i]$.[6] This procedure is summarized in Algorithm 2 (Section C). Notably, Corollary 2 shows that our trace is inherently NTT-friendly: the cost of the homomorphic trace is unchanged whether the input ciphertexts are in NTT form or not. This is important in practice, as it avoids expensive (I)NTT conversions.

Corollary 2 (Trace by Matrix Multiplication of NTT Components). *Let a, b be polynomials R'_q, and define polynomials b^* and c as in Theorem 3. Let the set of NTT matrices $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ over $\mathbb{Z}_q[i]$ as:*

$$\tilde{A}^{(\ell)}_{jk} = a(\zeta_j, \zeta_k, \eta_\ell), \tilde{B}^{(\ell)}_{jk} = b(\zeta_j, \zeta_k, \eta_\ell), \text{ and } \tilde{C}^{(\ell)}_{jk} = c(\zeta_j, \zeta_k, \eta_\ell),$$

[5] In our quotient ring, any element has a unique representation as a polynomial in Y^{-1}. We denote by $[Y^{-\ell}]g$ the coefficient g_ℓ in the unique expansion $g = \sum_{\ell=0}^{n-1} g_\ell Y^{-\ell}$.

[6] Equivalently, $2\varphi(p)$ matrix multiplications over $\mathbb{Z}_q$; cf. [27].

for ζ_j, ζ_k being the $4n$-th roots of unity and η_ℓ being the p-th roots of unity in $\mathbb{Z}_q$. Then, we have

$$\tilde{C}^{(\ell)} = \tilde{A}^{(\ell)} \tilde{B}^{(\ell)*}.$$

Proof. We can apply Lemma 5 on the set of distinct roots $\Omega = \{\zeta_\ell | \ell \in [0, n)\}$ of $Z^n - i$ over $\mathbb{Z}_q$, to obtain:

$$\mathrm{Tr}_Z\left(a(X, Z, W) \cdot b^*(Y, Z, W)\right) = \textstyle\sum_{\ell=0}^{n-1} a(X, \zeta_\ell, W) \cdot b^*(Y, \zeta_\ell, W).$$

The j, k, ℓ-th NTT component of c is given by:

$$\begin{aligned} c(\zeta_j, \zeta_k, \eta_\ell) &= \textstyle\sum_{\iota=0}^{n-1} a(\zeta_j, \zeta_\iota, \eta_\ell) \cdot \overline{b}(\zeta_k^{-1}, \zeta_\iota^{-1}, \eta_\ell^{-1}) \\ &= \textstyle\sum_{\iota=0}^{n-1} \tilde{A}^{(\ell)}_{j\iota} \cdot \overline{\tilde{B}^{(\ell)}_{k\iota}} = \left[\tilde{A}^{(\ell)} \tilde{B}^{(\ell)\mathrm{H}}\right]_{jk}. \end{aligned}$$

□

3.3 Auxiliary Ring and Non-power-of-two Matrix Arithmetic

We now extend the GL scheme to support non-power-of-two matrix arithmetics. The non-power-of-two matrix arithmetic is interesting in its own right, but it is also a crucial building block in our proposed bootstrapping algorithm. We consider the following ring:

$$R^{\mathsf{aux}} = \mathbb{Z}[i][X, W, V] / \langle X^n - i, \Phi_p(W), \Phi_p(V) \rangle.$$

Theorem 4 shows that matrix multiplication is well-defined using Tr_U over $R^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$.

Theorem 4 (Batched matrix multiplication in R^{aux}). *Let $a, b \in R^{\mathsf{aux}}$, and consider them as elements in $R^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$. Let*

$$c(X, W, V) = \mathrm{Tr}_U\left(a(X, W, U) \cdot \overline{b}(X^{-1}, V^{-1}, U^{-1})\right),$$

where $a(\zeta_\ell, \eta_j, \eta_k) = A^{(\ell)}_{jk}$ and $b(\zeta_\ell, \eta_j, \eta_k) = B^{(\ell)}_{jk}$, where matrices $A^{(\ell)}_{jk}, B^{(\ell)}_{jk} \in \mathcal{M}_{\varphi(p)}(\mathbb{C})$ for $\ell = 0, \ldots, n-1$. Then the following holds:

$$c(\zeta_\ell, \eta_j, \eta_k) = \left[A^{(\ell)} B^{(\ell)\mathrm{H}}\right]_{jk}.$$

Proof. Similarly to Theorem 2, we apply Lemmas 4 and 5 over $R^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$ to obtain with the decoding map $\Psi_{\ell,j,k} : R^{\mathsf{aux}} \to \mathbb{C}, a \mapsto a(\zeta_\ell, \eta_j, \eta_k)$:

$$\begin{aligned} \Psi_{\ell,j,k}(c) &= \mathrm{Tr}_U\left(a(\zeta_\ell, \eta_j, U) \cdot \overline{b}(\zeta_\ell^{-1}, \eta_k^{-1}, U^{-1})\right) \\ &= \textstyle\sum_{\iota=0}^{\varphi(p)-1} a(\zeta_\ell, \eta_j, \eta_\iota) \cdot \overline{b}(\zeta_\ell^{-1}, \eta_k^{-1}, \eta_\iota^{-1}) && \text{by Lemma 5} \\ &= \textstyle\sum_{\iota=0}^{\varphi(p)-1} A^{(\ell)}_{j\iota} \cdot \overline{B^{(\ell)}_{k\iota}} = \left[A^{(\ell)} B^{(\ell)\mathrm{H}}\right]_{jk}. \end{aligned}$$

□

Unlike the $Z^n - i$ case, the power-trace over $\Phi_p(U)$ introduces an additional correction term. Nevertheless, it can be computed as one square matrix multiplication plus a *rank-one outer product*, i.e., $O(n^2)$ time complexity.

Theorem 5. *Let $a(X,W,V), b(X,W,V)$ be polynomials in R^{aux} defined by prime p, and define the involution $b^*(X,W,V) := \bar{b}(X^{-1},W^{-1},V^{-1})$. Define matrices $\hat{A}$ and $\hat{B}'$ in $\mathcal{M}_{\varphi(p)}(R)$ by arranging the W,V-coefficients (with coefficients in $\mathbb{Z}[i][X]/\langle X^n - i\rangle$) as follows:*

$$[\hat{A}]_{j\ell} = [W^j V^\ell]a, \qquad [\hat{B}']_{k\ell} = [W^k V^{-\ell}]b^*,$$

where the entries are in $\mathbb{Z}[i][X]/\langle X^n - i\rangle$.

Then, for $c = \mathrm{Tr}_U\left(a(X,W,U)\cdot b^*(X,V,U)\right)$, the coefficient of W^jV^k is given by:

$$[W^jV^k]c = p\cdot[\hat{C}]_{jk} - [\hat{A}\mathbf{1}]_j\cdot[\hat{B}'\mathbf{1}]_k = [p\cdot\hat{C} - (\hat{A}\mathbf{1})\cdot(\hat{B}'\mathbf{1})^{\mathrm{T}}]_{jk},$$

where $\hat{C} = \hat{A}\hat{B}'^{\mathrm{T}}$ and $\mathbf{1}$ is the all-one vector of length $\varphi(p)$.

Proof. This theorem is proved using Lemma 2 in a way similar to the proof of Theorem 3; see Section B for details.

The first term is a matrix multiplication and costs $O(\varphi(p)^3)$, whereas the second term $(\hat{A}\mathbf{1})\cdot(\widehat{B'\mathbf{1}})^{\mathrm{T}}$ is a rank-one outer product and can be computed in $O(\varphi(p)^2)$. Hence the total complexity is asymptotically equivalent to that of plain matrix multiplication. Although Theorem 4 treats the case where p is prime, we can extend it to the odd prime-power case $p = r^k$ by applying Lemma 3. This introduces a rank-r^{k-1} term, which can be computed using a single matrix multiplication of a $\varphi(p)\times r^{k-1}$ matrix by a $r^{k-1}\times\varphi(p)$ matrix. Therefore, the total runtime remains $O(\varphi(p)^3)$.

This procedure is depicted in Algorithm 3 (Section C). It is worth noting that while the trace over non-NTT polynomials provides a well-structured formula, it becomes the same as $Z^n - i$ case in the NTT form.

Corollary 3 (Trace by Matrix Multiplication of NTT Form in Non-power-of-two Case). *Given a, b in R_q^{aux}, define b^* and c as in Theorem 5. Using the ring homomorphism $a \mapsto a(\zeta_\ell,\eta_j,\eta_k)$, define the set of NTT matrices $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ over $\mathbb{Z}_q[i]$ as:*

$$\tilde{A}^{(\ell)}_{jk} = a(\zeta_\ell,\eta_j,\eta_k), \tilde{B}^{(\ell)}_{jk} = b(\zeta_\ell,\eta_j,\eta_k), \text{ and } \tilde{C}^{(\ell)}_{jk} = c(\zeta_\ell,\eta_j,\eta_k),$$

where ζ_ℓ ranges over the $4n$-th roots of unity and η_j,η_k range over the p-th roots of unity in $\mathbb{Z}_q$. Then, we have

$$\tilde{C}^{(\ell)} = \tilde{A}^{(\ell)}\tilde{B}^{(\ell)\mathrm{H}}.$$

Proof. The claim follows directly from Lemma 5.

3.4 Encrypted Matrix Multiplication via Our Trace

In this subsection, we show that the trace operation defined above can be performed on ciphertexts in the GL scheme. Let $(b = m + e - a\cdot s, a) \in {R'_q}^2$ be a

ciphertext encrypting a plaintext $m \in R'$ under the secret key $s(X, W)$ (where s is constant with respect to Y), i.e.,

$$b + a \cdot s = m + e \pmod{q}.$$

We identify the (approximate) plaintext with the centered representative $[\![m+e]\!]$, simply denoted by $m+e$. Throughout the computation, we maintain $\|m+e\|_\infty < q/2$ to ensure decryption correctness.

Recall from Corollary 1 that modulo q reduction commutes with the trace operation. Therefore, we do not distinguish the notation between the trace over R' and that over R'_q. We define the ct–pt matrix multiplication using our trace operation as follows.

Theorem 6 (Homomorphic Trace (ct–pt) [27]). *Let $(b, a) \in R'^2_q$ be a ciphertext encrypting a plaintext $m \in R'$ under the secret key $s(X, W)$. Consider another plaintext $\mu \in R'_q$. Define polynomials in $R'_q[Z]/\langle \Phi_p(Z) \rangle$ as:*

$$d_0 = b(X, Z, W) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1}), \quad d_1 = a(X, Z, W) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1}).$$

Then, $(\mathrm{Tr}_Z(d_0), \mathrm{Tr}_Z(d_1)) \in R'^2_q$ is a ciphertext encrypting

$$\mathrm{Tr}_Z\left(m(X, Z, W) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1})\right)$$

being decrypted as follows using the secret key $s(X, W)$ and its permutation:

$$\mathrm{Tr}_Z(d_0) + \mathrm{Tr}_Z(d_1) \cdot s(X, W).$$

Proof. The proof directly follows from the decoding equation.

$$\begin{aligned}
\mathrm{Tr}_Z(d_0) + \mathrm{Tr}_Z(d_1) \cdot s(X, W) &= \mathrm{Tr}_Z\left(d_0 + d_1 \cdot s(X, W)\right) \qquad (\mathrm{Tr}_Z \text{ is } R'\text{-linear}) \\
&= \mathrm{Tr}_Z\left((m(X, Z, W) + e(X, Z, W)) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1})\right) \\
&= \left[\mathrm{Tr}_Z\left((m(X, Z, W) + e(X, Z, W)) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1})\right)\right]_q.
\end{aligned}$$

The correct decryption is ensured as long as

$$\| \mathrm{Tr}_Z\left((m(X, Z, W) + e(X, Z, W)) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1})\right) \|_\infty < q/2. \qquad \square$$

Lemma 6 (Noise Analysis of Homomorphic Trace). *Given polynomials m, μ in $R'[Z]/\langle Z^n - i \rangle$, assuming the use of the powerful basis, we have*

$$\| \mathrm{Tr}_Z(m \cdot \mu) \|_\infty \leq n \cdot \|m\|^{\mathsf{can}}_\infty \cdot \|\mu\|^{\mathsf{can}}_\infty,$$

where $\| \cdot \|^{\mathsf{can}}_\infty$ denotes the infinity norm of the canonical embedding.

Proof. We refer to Section B for the detailed proof.

Let $(\beta, \alpha) \in {R'_q}^2$ be another ciphertext encrypting a plaintext μ. The central task of GL scheme is to obtain a ciphertext encrypting the trace of the product, defined as:

$$\mathrm{Tr}_Z\left(m(X, Z, W) \cdot \overline{\mu}(Y^{-1}, Z^{-1}, W^{-1})\right).$$

We can also prove the ct–ct matrix multiplication using the linearity of the trace map.

Homomorphic Non-Power-of-Two Matrix Multiplication The above theorem generalizes to the auxiliary ring R^{aux} in a straightforward manner. Given a ciphertext $(b, a) \in (R_q^{\mathsf{aux}})^2$ encrypting a plaintext $m \in R^{\mathsf{aux}}$ under *the same secret key* $s(X, W)$, we can perform the homomorphic ct–pt matrix multiplication as follows. The following corollary is a direct extension of Theorem 6 and serves as the core building block for our bootstrapping algorithm.

Corollary 4 (ct–pt Matrix Multiplication, Extended). *Define polynomials in* $R_q^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$ *as:*

$$d_0 = b(X, W, U) \cdot \overline{\mu}(X^{-1}, V^{-1}, U^{-1}), \quad d_1 = a(X, W, V) \cdot \overline{\mu}(X^{-1}, V^{-1}, U^{-1}).$$

Then, $(\mathrm{Tr}_U(d_0), \mathrm{Tr}_U(d_1)) \in (R_q^{\mathsf{aux}})^2$ *is a ciphertext satisfying*

$$\mathrm{Tr}_U(d_0) + \mathrm{Tr}_U(d_1) \cdot s = \mathrm{Tr}_U\left(m(X, W, U) \cdot \overline{\mu}(X^{-1}, V^{-1}, U^{-1})\right).$$

Proof. The claim follows directly from the R_q^{aux}-linearity of the trace map Tr_U.

Lemma 7 (Noise Analysis of Homomorphic Trace, Extended). *Given polynomials* m, μ *in* $R^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$*, we have*

$$\| \mathrm{Tr}_U(m \cdot \mu) \|_\infty \leq \varphi(p) \cdot \|m\|_\infty^{\mathsf{can}} \cdot \|\mu\|_\infty^{\mathsf{can}}.$$

Proof. The proof is identical to that of Lemma 6, by replacing Tr_Z with Tr_U and working over $R^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$.

Given the second ciphertext $(\beta, \alpha) \in (R_q^{\mathsf{aux}})^2$ encrypting μ, we can compute a ciphertext encrypting $\mathrm{Tr}_U(m(X, W, U) \cdot \overline{\mu}(X^{-1}, V^{-1}, U^{-1}))$ by computing the trace of the products of the ciphertext components.

Corollary 5 (ct–ct Matrix Multiplication, Extended). *Let* $(b, a), (\beta, \alpha) \in (R_q^{\mathsf{aux}})^2$ *be ciphertexts encrypting plaintexts* $m, \mu \in R^{\mathsf{aux}}$ *under the same secret key* $s(X, W)$*. Define polynomials in* $R_q^{\mathsf{aux}}[U]/\langle \Phi_p(U) \rangle$ *as:*

$$d_0 = b(X, W, U) \cdot \overline{\beta}(X^{-1}, V^{-1}, U^{-1}), \quad d_1 = a(X, W, U) \cdot \overline{\beta}(X^{-1}, V^{-1}, U^{-1}),$$
$$d_2 = b(X, W, U) \cdot \overline{\alpha}(X^{-1}, V^{-1}, U^{-1}), \quad d_3 = a(X, W, U) \cdot \overline{\alpha}(X^{-1}, V^{-1}, U^{-1}).$$

Then, $(\mathrm{Tr}_U(d_0), \mathrm{Tr}_U(d_1), \mathrm{Tr}_U(d_2), \mathrm{Tr}_U(d_3)) \in (R_q^{\mathsf{aux}})^4$ *is a ciphertext encrypting*

$$\mathrm{Tr}_U\left(m(X, W, U) \cdot \overline{\mu}(X^{-1}, V^{-1}, U^{-1})\right)$$

being decrypted as follows using the secret key $s(X, W)$ *and its permutation:*

$$\mathrm{Tr}_U(d_0) + \mathrm{Tr}_U(d_1) \cdot s(X, W) + \mathrm{Tr}_U(d_2) \cdot \overline{s}(X^{-1}, V^{-1}) + \mathrm{Tr}_U(d_4) \cdot s(X, W)\overline{s}(X^{-1}, V^{-1}).$$

Matrix addition and the Hadamard product follow directly from the homomorphism $a \mapsto a(\zeta_\ell, \eta_k, \eta_j)$ and the commutativity of R_q^{aux}. The Galois action $X \mapsto X^{5^j}$ corresponds to an inter-matrix rotation by j steps, while $W \mapsto W^{\gamma^j}$ (resp., $V \mapsto V^{\gamma^j}$) corresponds to an intra-matrix row (resp., column) rotation by j steps. We can also realize transposition and Hermitian transposition as $(W, V) \mapsto (V, W)$ and $(i, X, W, V) \mapsto (-i, X^{-1}, V^{-1}, W^{-1})$, respectively. As explained in Section 2, these operations can be supported via proper key switching.

4 Efficient Bootstrapping for GL Scheme

This section assembles the ingredients from the previous sections into an efficient bootstrapping procedure for the GL scheme. Let $\mathsf{ct} = (b, a) \in R'^2_q$ be a ciphertext encrypting m under the secret key $s \in R_q$. By extracting coefficients along the Y-axis, we decompose ct into n independent ciphertexts

$$\mathsf{ct}_j = \big([Y^j]b, [Y^j]a\big) \in R_q^2, \quad j = 0, \ldots, n-1.$$

Since s is defined over R_q, each ct_j is a *valid ciphertext* encrypting $[Y^j]m$, namely

$$[Y^j]m \approx [Y^j]b + ([Y^j]a) \cdot s.$$

We bootstrap each ct_j independently. Our bootstrapping procedure follows the classical GHS procedure [24], especially its CKKS instantiation [14, 17]:

1. ModRaise: Using the centered representative with coefficients in $[-q/2, q/2)$, lift ct_j from modulus q to modulus $Q \gg q$.
2. CtS: Homomorphically encode (inverse DFT) the X- and W-axes into slots.
3. EvalMod: Perform approximate modulus reduction on each slot.
4. StC: Homomorphically decode (DFT) back to the coefficient representation in the X- and W-axes.

A key observation is that the (inverse) DFT is a linear transform and can be implemented via *matrix multiplication, a native operation in the GL scheme.* The DFT matrices for the X- and W-axes are given by

$$\mathcal{U}_X = [\zeta^{5^i \cdot j}]_{0 \le i,j < n} \quad \text{and} \quad \mathcal{U}_W = [\eta^{\gamma^i \cdot j}]_{0 \le i,j < \varphi(p)},$$

where 5 (resp. γ) is a generator of the corresponding power index set. The corresponding encoding (inverse DFT) matrices are given by $\mathcal{U}_X^{-1}$ and $\mathcal{U}_W^{-1}$. Hence, we utilize matrix-form DFTs for the CtS and StC steps.

The main challenge is to realize the required matrix multiplications along the X- and W-axes efficiently. In the GL scheme, the native ct–pt matrix multiplication multiplies the plaintext matrix on the *right* of the ciphertext (i.e., along the Y-axis). Moreover, implementing the multiplication along the W-axis requires an additional trick: we batch multiple ciphertexts using an auxiliary variable V.

4.1 ModRaise

Given a ciphertext ct_j over R_q, we first lift it to modulus Q by taking the centered representatives of its coefficients in $[-q/2, q/2)$. Over the larger ring R_Q (equivalently, before reduction modulo q), the lifted ciphertext satisfies

$$b + a \cdot s = m + e + qI \approx m + qI,$$

where I accounts for the ignored reduction modulo q. Under the standard RLWE assumption, one can derive a statistical bound on $\|I\|_\infty$; see [14, 9, 35] for details.

Our goal is to map the value back to its centered representative modulo q, i.e., to approximately evaluate the modular reduction $x \mapsto [x]_q$. This is essentially a coefficient-wise operation; however, since native homomorphic operations act slot-wise, we cannot directly apply this to the coefficients. Instead, we employ CtS, a homomorphic linear transform that maps coefficients to slot values, to evaluate $x \mapsto [x]_q$ slot-wise, and then decode the result via StC.

4.2 CoeffToSlot

Let $\mathsf{ct}_j^{\mathsf{mr}} \in R_Q^2$ denote the output of $\mathsf{ModRaise}$, and let m_j^{mr} be its underlying plaintext polynomial. Consider the partial decoding homomorphism $\Psi_{X,W} : R \to \mathbb{C}^{n\times\varphi(p)}, m_j^{\mathsf{mr}} \mapsto M^{\mathsf{mr}(j)} = [m_j^{\mathsf{mr}}(\zeta_\alpha, \eta_\beta)]_{0\le\alpha<n, 0\le\beta<\varphi(p)}$; the goal of CtS is to transform the ciphertext so that the slots *hold* the coefficient matrix:

$$[[X^\alpha W^\beta] m_j^{\mathsf{mr}}]_{0\le\alpha<n, 0\le\beta<\varphi(p)}.$$

This corresponds to performing the inverse DFT (encoding) along both the X- and W-axes. In matrix notation, we compute:

$$\mathcal{U}_X^{-1} M^{\mathsf{mr}(j)} {\mathcal{U}_W^{-1}}^{\mathrm{T}} = [[X^\alpha W^\beta] m_j^{\mathsf{mr}}]_{0\le\alpha<n, 0\le\beta<\varphi(p)}.$$

***X*-axis Inverse DFT** We first address the multiplication by $\mathcal{U}_X^{-1}$. The following theorem establishes a method for performing this matrix multiplication homomorphically using the trace operation.

Theorem 7 (Matrix Multiplication along X-axis). *Let $R = \mathbb{Z}[i][X,W]/\langle X^n - i, \Phi_p(W)\rangle$. Consider the partial decoding map*

$$\Psi_{X,W} : R \to \mathbb{C}^{n\times\varphi(p)}, \quad m \mapsto M = [m(\zeta_j, \eta_k)]_{0\le j<n, 0\le k<\varphi(p)}.$$

For a target matrix $A \in \mathcal{M}_n(\mathbb{C})$, let $a(X,Z) \in R[Z]/\langle Z^n - i\rangle$ be a polynomial independent of W, satisfying $a(\zeta_j, \zeta_k) = A_{jk}$ for all $j,k \in [0,n)$. Then, for $c(X,W) = \mathrm{Tr}_Z\left(a(X,Z)\cdot m(Z,W)\right)$, we have

$$\Psi_{X,W}(c) = AM,$$

where Tr_Z denotes the trace over the extension $R[Z]/\langle Z^n - i\rangle$.

The remaining challenge is that we cannot directly apply Tr_Z to $\mathsf{ct}_j^{\mathsf{mr}}$. Since the corresponding secret key $s(Z,W)$ depends on the variable Z, it does not commute with the trace operation Tr_Z. To resolve this, we first pack the ciphertexts along the Y-axis (which is independent of s) and then align the dimensions using Hermitian transposition.

Step 1: Packing and Transpose. We pack n of $\mathsf{ct}_j^{\mathsf{mr}} \in (R_Q)^2$ into $\mathsf{ct}^{\mathsf{ext.y}} \in (R'_Q)^2$:

$$\mathsf{ct}^{\mathsf{ext.y}} = \textstyle\sum_{j=0}^{n-1} \mathsf{ct}_j^{\mathsf{mr}} \cdot Y^j,$$

which encrypts the message $m^{\mathsf{ext.y}}(X,Y,W) = \sum_{j=0}^{n-1} m_j^{\mathsf{mr}}(X,W)\cdot Y^j$. We then perform the homomorphic Hermitian transpose on $\mathsf{ct}^{\mathsf{ext.y}}$, followed by key switching to $s(X,W)$. The resulting ciphertext, denoted $\mathsf{ct}^{\mathsf{tr.x}}$, encrypts the Hermitian transposed message $\overline{m^{\mathsf{ext.y}}}(Y^{-1}, X^{-1}, W^{-1})$.

Step 2: Homomorphic Trace. Given the encoding polynomial $u_X(X, Y)$ for $\mathcal{U}_X^{-1}$, we compute:

$$\mathsf{ct}^{\mathsf{cts.tr.x}} = \mathrm{Tr}_Z\left(\mathsf{ct}^{\mathsf{tr.x}}(X, Z, W) \cdot \overline{u_X}(Y^{-1}, Z^{-1})\right),$$

where the operations are applied component-wise to the ciphertext tuple. This results in a ciphertext encrypting:

$$m^{\mathsf{cts.tr.x}} = \mathrm{Tr}_Z\left(\overline{m^{\mathsf{ext.y}}}(Z^{-1}, X^{-1}, W^{-1}) \cdot \overline{u_X}(Y^{-1}, Z^{-1})\right).$$

Step 3: Transpose Back. Finally, we apply the Hermitian transpose on $\mathsf{ct}^{\mathsf{cts.tr.x}}$ to obtain $\mathsf{ct}^{\mathsf{cts.x}}$. Using the property $\mathrm{Tr}_Z(f(Z^{-1})) = \mathrm{Tr}_Z(f(Z))$, the underlying message becomes:

$$\begin{aligned} m^{\mathsf{cts.x}} &= \overline{m^{\mathsf{cts.tr.x}}}(Y^{-1}, X^{-1}, W^{-1}) = \mathrm{Tr}_Z\left(m^{\mathsf{ext}}(Z, Y, W) \cdot u_X(X, Z)\right) \\ &= \textstyle\sum_{j=0}^{n-1} \mathrm{Tr}_Z\left(m_j^{\mathsf{mr}}(Z, W) \cdot u_X(X, Z)\right) \cdot Y^j. \end{aligned}$$

By Theorem 7, each coefficient of Y^j in $\mathsf{ct}^{\mathsf{cts.x}}$ corresponds to the ciphertext of the desired matrix:

$$\mathcal{U}_X^{-1} \cdot M^{\mathsf{mr}(j)}.$$

Extracting these coefficients yields the X-axis encoded ciphertexts.

W-axis Inverse DFT We now present the *core integration of our algebraic framework*. The concepts established thus far—the generalized trace, power automorphisms, and the auxiliary ring R_q^{aux}—are combined to enable efficient homomorphic encoding along the W-axis. This synthesis overcomes the structural limitation of the GL scheme; while the scheme supports native matrix multiplication along the Y-axis, it offers no direct primitive for multiplication in the W-direction.

The W-axis encoding follows a three-step procedure analogous to the X-axis case, but involves an auxiliary packing step using the variable V.

Theorem 8. *Define R and $\Psi_{X,W}$ as in Theorem 7. For a target matrix $A \in \mathcal{M}_{\varphi(p)}(\mathbb{C})$, let $a(W, U) \in \mathbb{Z}_q[i][W, U]/\langle \Phi_p(W), \Phi_p(U)\rangle$ be a polynomial satisfying the interpolation condition: $a(\eta_j, \eta_k) = A_{jk}$ for all $j, k \in [0, \varphi(p))$. Then, for $c(X, W) = \mathrm{Tr}_U\left(a(W, U) \cdot m(X, U)\right)$, we have*

$$\Psi_{X,W}(c) = MA^{\mathrm{T}},$$

where Tr_U denotes the trace over the extension $R[U]/\langle \Phi_p(U)\rangle$.

As before, we cannot directly apply Tr_U to the lifted ciphertexts because the secret key $s(X, U)$ depends on the variable U. To address this, we perform batched decoding by introducing an auxiliary variable V.

Step 1: Packing and Transpose. We begin with n independent ciphertexts $\mathsf{ct}_j^{\mathsf{cts.x}} \in R_Q^2$ for $j = 0, \ldots, n-1$. For simplicity, we first assume $\varphi(p) = n$; the general case is addressed subsequently. We pack these ciphertexts into a single ciphertext $\mathsf{ct}^{\mathsf{ext.v}} \in (R_Q^{\mathsf{aux}})^2$:

$$\mathsf{ct}^{\mathsf{ext.v}} = \textstyle\sum_{j=0}^{n-1} \mathsf{ct}_j^{\mathsf{cts.x}} \cdot V^j,$$

which encrypts the message $m^{\mathsf{ext.v}}(X, W, V) = \sum_{j=0}^{n-1} m_j^{\mathsf{cts.x}}(X, W) \cdot V^j$.

We perform the homomorphic Hermitian transpose on $\mathsf{ct}^{\mathsf{ext.v}}$, followed by key switching from $\overline{s}(X^{-1}, V^{-1})$ to $s(X, W)$. The Hermitian transposed ciphertext $\mathsf{ct}^{\mathsf{tr.w}}$ encrypts $\overline{m^{\mathsf{ext}}}(X^{-1}, V^{-1}, W^{-1})$.

Step 2: Homomorphic Trace over $R_q^{\mathsf{ext}}[U]/\langle \Phi_p(U) \rangle$. Given the encoding polynomial $u_W(W, U)$ for $\mathcal{U}_W^{-1}$, we compute:

$$\mathsf{ct}^{\mathsf{cts.tr.w}} = \mathrm{Tr}_U \left(\mathsf{ct}^{\mathsf{tr.w}}(X, W, U) \cdot \overline{u_W}(V^{-1}, U^{-1}) \right).$$

The corresponding plaintext polynomial becomes:

$$m^{\mathsf{cts.tr.w}} = \mathrm{Tr}_U \left(\overline{m^{\mathsf{ext}}}(X^{-1}, U^{-1}, W^{-1}) \cdot \overline{u_W}(V^{-1}, U^{-1}) \right).$$

Step 3: Transpose Back. Finally, we apply the reverse Hermitian transpose using the map $(i, X, W, V) \mapsto (-i, X^{-1}, V^{-1}, W^{-1})$. As in the X-axis case, using the property $\mathrm{Tr}_U(f(U^{-1})) = \mathrm{Tr}_U(f(U))$, the underlying message becomes:

$$m^{\mathsf{cts}} = \mathrm{Tr}_U \left(m^{\mathsf{ext.v}}(X, U, V) \cdot u_W(W, U) \right).$$

We then extract each coefficient of V^j to obtain the W-axis encoded ciphertexts. Specifically, we obtain $\varphi(p)$ ciphertexts $\mathsf{ct}_j^{\mathsf{cts}}$, each encrypting:

$$\begin{aligned} [V^j] m^{\mathsf{cts}} &= \mathrm{Tr}_U \left([V^j] m^{\mathsf{ext.v}}(X, U, V) \cdot u_W(W, U) \right) \\ &= \mathrm{Tr}_U \left(m^{\mathsf{mr}}{}_j(X, U) \cdot u_W(W, U) \right), \qquad \because m^{\mathsf{ext.v}} = \textstyle\sum_{j=0}^{n-1} m_j^{\mathsf{cts.x}} \cdot V^j \end{aligned}$$

which, by Theorem 8, corresponds to the desired right-multiplication by ${\mathcal{U}_W^{-1}}^{\mathrm{T}}$. Consequently, we obtain ciphertexts encrypting:

$$\mathcal{U}_X^{-1} \cdot M^{\mathsf{mr}(j)} \cdot {\mathcal{U}_W^{-1}}^{\mathrm{T}}.$$

More generally, depending on the choice of parameters, $\varphi(p)$ may not equal n. We handle the possible cases as follows:

1. Case $\varphi(p) = n$: Described above.
2. Case $\varphi(p) > n$: Pad with trivial ciphertexts (i.e., $(0, 0)$) upto $\varphi(p)$.
3. Case $\varphi(p) < n$: Divide the inputs into $\lceil n/\varphi(p) \rceil$ groups and do the same.

Alternatively, we can batch k ciphertexts in R'_q for bootstrapping, yielding $k \cdot m$ independent ciphertexts in R_q. In this scenario, we require $\lceil km/\varphi(p) \rceil$ homomorphic traces along the W-axis.

4.3 EvalMod

We perform approximate modulus reduction on each ciphertext. Conceptually, this step corresponds to the modular reduction in CKKS bootstrapping via a sine approximation (and its variants). In this step, ciphertexts reside in the base ring R (rather than in extension or auxiliary rings), and $\Psi_{X,W}$ implements a full DFT across all slots [17, 39]. We therefore use the efficient switching key $\mathsf{Switch}_{\mathsf{small}}$ as the relinearization key. Moreover, the polynomial evaluation for the n independent ciphertexts can be fully parallelized.

4.4 SlotToCoeff

The StC step essentially inverts CtS. Using the same techniques as in CtS—transposition and homomorphic traces—we apply the DFT along the W- and X-axes. For each ciphertext whose underlying plaintext is M^{mod}, we compute

$$\mathcal{U}_X \cdot M^{\mathsf{mod}} \cdot \mathcal{U}_W^{\mathrm{T}}.$$

4.5 Complexity Analysis

We analyze the runtime by separating (i) key-switching overhead and (ii) plaintext (matrix) multiplication. This separation is important for two reasons. First, our construction reduces the dominant arithmetic to *plain* matrix multiplication, which is straightforward to optimize with standard libraries and cache-friendly implementations, whereas prior HE approaches typically require carefully structured sequences of operations that do not admit the same low-level optimizations; hence, a single end-to-end bound can be misleading. Second, key switching is often governed not only by arithmetic complexity but also by memory movement (switching-key bandwidth), so reporting it separately yields a more faithful comparison across schemes.

Key Switching Cost Our CtS (and similarly StC) consists of four key switchings and two homomorphic traces. Each key switching costs either $O(n^2 p \log(np))$ (for the X-axis) or $O(\lceil n/p \rceil \cdot np^2 \log(np))$ (for the W-axis); without loss of generality, assume $n \geq p$. Therefore, the total key-switching complexity is

$$O(n^2 p \log(np)).$$

Dividing by the number of slots $n^2\varphi(p)$, the amortized key-switching cost is

$$O((p/\varphi(p)) \log(np)) = O(\log(np)).$$

For comparison, the best-known CtS algorithm in CKKS/BGV/BFV schemes is based on large-radix FFT techniques [13, 9]. Let δ be the FFT depth. At each stage, one multiplies by a matrix of size $\sqrt[\delta]{n}$, which requires $O(n^{1/2\delta})$ rotations using the double-hoisting technique. Each rotation incurs a key switching of cost $O(n \log n)$. Hence, the total key-switching complexity is $O(\delta \cdot n^{1+1/2\delta} \log n)$. Dividing by n, the amortized key-switching cost is $O(\delta \cdot n^{1/2\delta} \log n)$, where δ is chosen as large as possible without consuming too many levels; a practical choice is $\delta \in \{2, 3, 4\}$.

Plaintext Multiplication Cost The plaintext matrix-multiplication cost is approximately $O(n^3 p)$ for the X-axis and $O(\lceil n/p \rceil \cdot np^3)$ for the W-axis, resulting in a total complexity of $O(n^3 p + np^3)$. Dividing by $n^2\varphi(p)$, the amortized plaintext-multiplication cost becomes $O(n + p^2/n)$. This cost is minimized when $n \approx p$; hence, given the total number of slots $N = n \cdot \varphi(p) \geq 2^{15}$, the amortized complexity scales as $O(\sqrt{N})$.

Large-radix FFT-based CtS methods for CKKS/BGV/BFV [13, 9] perform multiplication by a vector of (encoded) twiddle factors at each stage. For an FFT depth of δ, this requires $O(N^{1/\delta})$ plaintext multiplications per stage, where N is the total number of slots. Consequently, the total plaintext-multiplication complexity is $O(\delta \cdot N^{1+1/\delta})$, and the amortized cost is $O(\delta \cdot N^{1/\delta})$. Theoretically, the plaintext multiplication cost of prior schemes is cheaper when $\delta > 2$.

5 Implementation Results

5.1 Optimizations

Although our algorithm significantly reduces the number of key-switching operations during CtS and StC, it also allows us to further optimize the computation by achieving an even lower number of key-switching operations.

Reducing Two Key Switchings At a high level, CtS applies homomorphic encoding along the X- and the W-axes, which requires four key switchings to perform the Hermitian transposition on the ciphertext. By partitioning the ciphertext along the X-axis instead of the Y-axis, the initial key switching in the X-axis inverse DFT phase can be removed. Under this modification, the ciphertext can be viewed as having implicitly undergone the Hermitian transposition. Consequently, CtS and StC may be interpreted as performing homomorphic encoding and decoding along the Y- and the W-axes, respectively. This optimization reduces the total number of key switchings from four to three. Moreover, since the two matrix multiplications ($\mathcal{U}_X$ and $\mathcal{U}_W$) within each of CtS and StC commute, we may remove the key switching on either axis, depending on performance.

Sparse-Secret Encapsulation and Lazy Key-Switching The sparse-secret encapsulation technique [10] can be applied to the GL scheme to reduce both the failure probability and the computational overhead of EvalMod. In this approach, a ciphertext is temporarily key-switched from the original secret key to a sparse secret key s' with low Hamming weight h. The ModRaise operation is then applied to increase the modulus, after which the ciphertext is key-switched back to the original secret key. As a result, ℓ_∞ norm of I term produced by ModRaise is reduced, since $\|I\|_\infty \leq 1.81\sqrt{h}$ with high probability [9, 10]. Because the cost of EvalMod for removing I depends directly on $\|I\|_\infty$, this technique leads to a reduction in computational cost.

Algorithm 1 Optimization for $\mathsf{ModRaise}$ and CtS

Input: $\mathsf{ct}_s \in R'^2_{q_0}$, $\mathsf{ksk}_{s\mapsto s'}$, $\mathsf{ksk}_{s'^*\mapsto s}$, $\mathsf{ksk}^{\mathsf{aux}}_{s^*\mapsto s}$
Output: $\mathsf{ct}^{\mathsf{cts}}_{s,j} \in R^2_{Q_{L-1}}$ for $j = 0, \ldots, n-1$

$\mathsf{ct}_{s'} \leftarrow \mathsf{Switch}_{\mathsf{small}}(\mathsf{ct}_s, \mathsf{ksk}_{s\mapsto s'})$
for $j = 0$ to $n-1$ **do**
  $\mathsf{ct}^{\mathsf{mr}}_{s',j} \leftarrow \mathsf{ModRaise}(\mathsf{ct}_{s',j})$ ▷ ModRaise to $Q_L P$
$\mathsf{ct}^{\mathsf{tr.x.sparse}}_{s'} \leftarrow \sum_{j=0}^{n-1} \mathsf{ct}^{\mathsf{mr}}_{s',j} \cdot Y^j$ ▷ $\mathsf{ct}^{\mathsf{tr.x.sparse}}_{s'} \in (R'_{Q_L P})^2$
$\mathsf{ct}^{\mathsf{cts.tr.x.sparse}}_{s'} \leftarrow \mathsf{rescale}(\mathrm{Tr}_Z\left(\mathsf{ct}^{\mathsf{tr.x.sparse}}_{s'}(X, Z, W) \cdot \overline{u}_X(Y^{-1}, Z^{-1})\right))$ ▷ Rescale to Q_L
$\mathsf{ct}^{\mathsf{cts.x}}_s \leftarrow \mathsf{Herm}(\mathsf{ct}^{\mathsf{cts.tr.x.sparse}}_{s'}, \mathsf{ksk}_{s'^*\mapsto s})$
$\mathsf{ct}^{\mathsf{ext.v}}_s \leftarrow \sum_{j=0}^{\varphi(p)-1} \mathsf{ct}^{\mathsf{cts.x}}_{s,j} \cdot V^j$ ▷ $\mathsf{ct}^{\mathsf{ext.v}}_s \in (R^{\mathsf{aux}}_{Q_L})^2$
$\mathsf{ct}^{\mathsf{tr.w}}_s \leftarrow \mathsf{Herm}(\mathsf{ct}^{\mathsf{ext.v}}_s, \mathsf{ksk}^{\mathsf{aux}}_{s^*\mapsto s})$
$\mathsf{ct}^{\mathsf{cts.tr.w}}_s \leftarrow \mathsf{rescale}(\mathrm{Tr}_U\left(\mathsf{ct}^{\mathsf{tr.w}}_s(X, W, U) \cdot \overline{u}_W(V^{-1}, U^{-1})\right))$ ▷ Rescale to Q_{L-1}
$\mathsf{ct}^{\mathsf{cts}}_s \leftarrow \mathsf{Herm}(\mathsf{ct}^{\mathsf{cts.tr.w}}_s, \mathsf{ksk}^{\mathsf{aux}}_{s^*\mapsto s})$
return $\left\{\mathsf{ct}^{\mathsf{cts}}_{s,j}\right\}_{j=0}^{n-1}$

In GL scheme, the second key switching can be performed lazily and merged with the first Hermitian transpose in CtS, thereby eliminating an additional key-switching. While lazy key-switching can also be applied to CKKS bootstrapping, its impact is limited because CtS in CKKS involves multiple key-switching operations. In contrast, eliminating a single key switching in the GL scheme yields a significant performance improvement.

$\mathsf{ModRaise}$ to QP Combining the above two optimizations enables further reductions in level consumption. In particular, one can directly compute the homomorphic trace of the product of the mod-raised ciphertext and the encoding map without requiring any key switching. This approach allows the ciphertext to be modulus raised to modulus QP, where P is an additional prime, and then rescaled by P after the matrix multiplication. As a result, one level is saved during the matrix multiplication step.

Putting It Together Algorithm 1 summarizes the optimized procedure. Let $Q_L = q_L q_{L-1} \cdots q_0$, where $\{q_i\}_{i=0}^{L}$ are the primes up to level L. Let u_X and u_W denote the encoded inverse DFT matrices along the X- and W-axes, respectively, and assume that u_X is encoded with scaling factor P. For simplicity we assume $\varphi(p) = n$; the algorithm extends to the general case. Let s' be a sparse secret of lower Hamming weight. The algorithm uses three key-switching keys: $\mathsf{ksk}_{s\mapsto s'} \in R^{2d_g}$, $\mathsf{ksk}_{s'^*\mapsto s} \in R'^{2d_g}$, and $\mathsf{ksk}^{\mathsf{aux}}_{s^*\mapsto s} \in (R^{\mathsf{aux}})^{2d_g}$.

The output of Algorithm 1 consists of n ciphertexts that are homomorphically encoded along both Y- and W-axes. After applying $\mathsf{EvalMod}$, the StC can be implemented by applying the above steps in reverse order. As a result, the final Hermitian transposition can be omitted.

Table 1. Parameters for experiments. "Coeff Count" denotes the total number of ring coefficients. n and p are the degrees in X and W, respectively. For Lattigo (CKKS), n corresponds to a length-n vector; our setting is roughly comparable to processing 64 batched CKKS ciphertexts with $\log_2 N = 15$. h and $\tilde{h}$ are the Hamming weights for standard ciphertexts and sparse-secret encapsulation. Q is the ciphertext modulus, and QP is the extended modulus for key switching.

Set	Coeff Count	n	p	h	$\tilde{h}$	$\log_2 Q$	$\log_2 QP$
Ours	$2 \times 64 \times 64 \times 256$	64	257	Uniform	128	748	868
I	32768	16384	-	192	32	746	868
II	65536	32768	-	192	32	1416	1721

Table 2. Bootstrapping performance comparison. The column "Slot Count" denotes the number of slots in a single ciphertext, and "Batch Count" indicates the number of ciphertexts processed simultaneously during batched bootstrapping. $\log_2 Q'$ is the output coefficient modulus size after bootstrapping, and $\log_2 \epsilon^{-1}$ is the output precision.

	Ours			**I**		**II**	
	Time (ms)	Amort. (ms)	%	Time (ms)	%	Time (ms)	%
ModRaise	3,374	13.18	0.2	143.82	2.8	989.47	5.7
CoeffToSlot	288,912	1,128.56	17.2	3,224.50	63.2	8,587.48	49.7
EvalMod	1,338,951	5,230.28	79.7	1,325.54	26.0	6,803.75	39.3
SlotToCoeff	48,720	190.31	2.9	410.41	8.0	910.69	5.3
Total	1,679,957	6,562.33	100	5,104.27	100	17,291.39	100
Slot Count	$64 \times 64 \times 256$			16384		32768	
Batch Count	4			1		1	
$\log_2 Q'$	100			46		900	
$\log_2 \epsilon^{-1}$	17.61			13.53		19.49	

Table 3. Runtime for each operation in CtS and StC. The symbols R' and R^{aux} denote the ring spaces in which the operations are performed. Count indicates the number of times each operation is executed during CtS and StC.

		CtS		StC	
		Transpose	Matrix Mult.	Transpose	Matrix Mult.
$R'(\mathcal{U}_X)$	Time (ms)	92,595	21,100	14,192	6,000
	Count	8	4	8	4
$R^{\mathsf{aux}}(\mathcal{U}_W)$	Time (ms)	82,480	39,935	11,137	8,065
	Count	1	1	1	1

5.2 Parameter Selection and Performance

We follow the security guidelines of Bossuat et al. [8] to achieve 128-bit security under the RLWE assumption with ternary secrets. The resulting parameter sets used in our experiments are summarized in Table 1. The row labeled "Ours" corresponds to the GL scheme, while Sets I and II correspond to the CKKS scheme using the open-source library Lattigo [1].

For the matrix multiplication, we use the FLINT library [21] to implement modular matrix multiplication. All experiments were conducted on a machine

equipped with an Intel Core i7-10700K CPU @ 5.10 GHz in single-threaded mode and 64 GB of RAM. The performance comparison is presented in Table 2. For EvalMod, we use a degree-49 polynomial approximation, whereas Sets I and II use degree 30; thus, a direct one-to-one comparison should be interpreted with care. The runtime of the Hermitian transposition including key switching and ct–pt matrix multiplication is summarized in Table 3.

Result Interpretation As shown in Table 2, our technique achieves approximately a 3× throughput improvement for both CtS and StC compared to parameter set I. In addition, the overhead of ModRaise is reduced to just 0.2% of the total runtime in our implementation due to the elimination of one key-switching step during sparse-secret encapsulation.

In terms of overall runtime, our implementation is slower than Lattigo. However, the current implementation serves as a proof of concept and has not been fully optimized. There also exists a discrepancy in the choice of h, which negatively affects performance. Since EvalMod itself remains unchanged while the linear part is already faster in our approach, we expect that, after standard engineering optimizations and parameter alignment, the overall bootstrapping runtime should be faster than that of Lattigo.

As shown in Table 3, a single matrix multiplication is more than 2× faster than a single transpose operation in CtS. Observe that operations in R^{aux} are approximately 8 times slower than those in R', even though the number of coefficients differs by only factor of 4 (the number of coefficients in R^{aux} is $64 \times 256 \times 256$, whereas in R' it is $64 \times 64 \times 256$). This arises because the NTT over non-power-of-two dimensions is implemented using Rader's FFT algorithm [42], which internally performs two NTTs over power-of-two dimensions, thereby increasing the computational cost. Given that key switching is theoretically cheaper than matrix multiplication, our results indicate substantial room for optimization. The current overhead largely stems from a generic, non-specialized implementation, and we expect that tailored parameter choices and low-level arithmetic optimizations would significantly reduce this cost.

6 Conclusion

In this work, we generalized the GL scheme to support matrix arithmetic over non-power-of-two dimensions and introduced streamlined CtS and StC procedures that require only $O(1)$ key-switching operations. By reformulating the bootstrapping steps as ct–pt matrix multiplications, we eliminated the complicated rotations and masked twiddle-factor multiplications used in prior work. This approach simplifies the circuit and offers substantial performance benefits. Our experiments show that the CtS/StC runtime is much faster than the EvalMod step. It is worth noting that despite the lack of low-level optimizations in our proof-of-concept implementation, it already outperforms the well-optimized Lattigo baseline in the CtS/StC phases.

The theoretical foundation of our method lies in the properties of the trace over commutative rings. In revisiting the definitions and theorems of the GL scheme, we identified a practically important property: the trace in a GL ciphertext can be computed simply via matrix multiplication, regardless of whether the ciphertext is in NTT form. This property is unlikely to hold in coefficient-encoding-based methods [41, 6, 15]. Consequently, this flexibility allows for efficient trace evaluation in $R[Z]/\langle \Phi_p(Z) \rangle$ for odd prime powers p.

Furthermore, we highlight a fundamental structural difference between GL and conventional SIMD schemes. While schemes like BGV, BFV, and CKKS rely on encoding homomorphisms that map directly to component-wise addition and multiplication, the GL scheme introduces a distinct homomorphism: *lift to an extended algebra, permute, multiply in the extension, and trace back.* This suggests that the space of useful homomorphic operations may be richer than what is captured by addition and multiplication alone, and motivates the search for further primitives of this kind for new algorithmic designs in HE.

Despite these advances, EvalMod remains the primary bottleneck in bootstrapping due to the significant depth consumption of approximate polynomial evaluations, which necessitates the use of large parameters. Currently, we lack a direct method for homomorphic modular reduction. Just as the GL scheme leverages native matrix arithmetic to bypass previous limitations, an *inside-out* approach to modular reduction appears inherently more challenging, but could yield substantial performance gains if it can be realized. Additionally, the asymptotic complexity of our CtS/StC procedures is currently dominated by matrix multiplication. Future work will focus on identifying FFT-like structures in the encoding matrices $\mathcal{U}_X$ and $\mathcal{U}_W$ to reduce the asymptotic complexity from $O(n^3\varphi(p)+n\varphi(p)^3)$ to $O((n^2\varphi(p)+n\varphi(p)^2)\log(n\varphi(p)))$. Finally, given the heavy reliance of transformer models on matrix multiplication and its circuit depth, applying the generalized GL scheme and our bootstrapping to privacy-preserving transformers stands out as a primary direction for further investigation.

Acknowledgments The authors used AI tools (ChatGPT and Gemini) for grammatical refinement and to assist in generating LaTeX code for tables and TikZ figures. All final content was verified by the authors.

References

1. Lattigo v6. Online: https://github.com/tuneinsight/lattigo (Aug 2024), ePFL-LDS, Tune Insight SA
2. Aikata, A., Roy, S.S.: Secure and efficient outsourced matrix multiplication with homomorphic encryption. In: Mukhopadhyay, S., Stănică, P. (eds.) INDOCRYPT 2024, Part I. LNCS, vol. 15495, pp. 51–74. Springer, Cham, Switzerland, Chennai, India (Dec 18–21, 2024). https://doi.org/10.1007/978-3-031-80308-6_3
3. Alexandru, A., Kim, A., Polyakov, Y.: General functional bootstrapping using CKKS. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part III. LNCS,

vol. 16002, pp. 304–337. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2025). https://doi.org/10.1007/978-3-032-01881-6_10
4. Bae, Y., Cheon, J.H., Cho, W., Kim, J., Kim, T.: META-BTS: Bootstrapping Precision Beyond the Limit. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 223–234. ACM, Los Angeles CA USA (Nov 2022). https://doi.org/10.1145/3548606.3560696
5. Bae, Y., Cheon, J.H., Cho, W., Kim, J., Kim, T.: META-BTS: Bootstrapping precision beyond the limit. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 223–234. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). https://doi.org/10.1145/3548606.3560696
6. Bae, Y., Cheon, J.H., Hanrot, G., Park, J.H., Stehlé, D.: Plaintext-ciphertext matrix multiplication and FHE bootstrapping: Fast and fused. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part III. LNCS, vol. 14922, pp. 387–421. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68382-4_12
7. Bae, Y., Cheon, J.H., Kim, J., Stehlé, D.: Bootstrapping bits with CKKS. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 94–123. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58723-8_4
8. Bossuat, J.P., Cammarota, R., Chillotti, I., Curtis, B.R., Dai, W., Gong, H., Hales, E., Kim, D., Kumara, B., Lee, C., Lu, X., Maple, C., Pedrouzo-Ulloa, A., Player, R., Polyakov, Y., Lopez, L.A.R., Song, Y., Yhee, D.: Security guidelines for implementing homomorphic encryption. Cryptology ePrint Archive, Paper 2024/463 (2024), https://eprint.iacr.org/2024/463
9. Bossuat, J.P., Mouchet, C., Troncoso-Pastoriza, J.R., Hubaux, J.P.: Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 587–617. Springer, Cham, Switzerland, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77870-5_21
10. Bossuat, J.P., Troncoso-Pastoriza, J.R., Hubaux, J.P.: Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In: Ateniese, G., Venturi, D. (eds.) ACNS 2022. LNCS, vol. 13269, pp. 521–541. Springer, Cham, Switzerland, Rome, Italy (Jun 20–23, 2022). https://doi.org/10.1007/978-3-031-09234-3_26
11. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012). https://doi.org/10.1007/978-3-642-32009-5_50
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM, Cambridge, MA, USA (Jan 8–10, 2012). https://doi.org/10.1145/2090236.2090262
13. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 34–54. Springer, Cham, Switzerland, Darmstadt, Germany (May 19–23, 2019). https://doi.org/10.1007/978-3-030-17656-3_2
14. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 360–384. Springer, Cham, Switzerland, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78381-9_14

15. Cheon, J.H., Kang, M., Lee, J.: Fast batch matrix multiplication in ciphertexts. Cryptology ePrint Archive, Report 2025/1957 (2025), https://eprint.iacr.org/2025/1957
16. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham, Switzerland, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70694-8_15
17. Cheon, J.H., Kim, A., Yhee, D.: Multi-dimensional packing for HEAAN for approximate matrix arithmetics. Cryptology ePrint Archive, Report 2018/1245 (2018), https://eprint.iacr.org/2018/1245
18. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 377–408. Springer, Cham, Switzerland, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70694-8_14
19. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer Berlin Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46800-5_24
20. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
21. FLINT: Fast Library for Number Theory (2025), version 3.3.1, https://flintlib.org
22. Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford University (2009), https://crypto.stanford.edu/craig/, PhD thesis
23. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 169–178. ACM Press, Bethesda, MD, USA (May 31 – Jun 2, 2009). https://doi.org/10.1145/1536414.1536440
24. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 1–16. Springer Berlin Heidelberg, Germany, Darmstadt, Germany (May 21–23, 2012). https://doi.org/10.1007/978-3-642-30057-8_1
25. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer Berlin Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4_28
26. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012). https://doi.org/10.1007/978-3-642-32009-5_49
27. Gentry, C., Lee, Y.: Fully homomorphic encryption for matrix arithmetic. Cryptology ePrint Archive, Report 2025/1935 (2025), https://eprint.iacr.org/2025/1935
28. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_5
29. Greither, C.: Cyclic Galois Extensions of Commutative Rings, Lecture Notes in Mathematics, vol. 1534. Springer-Verlag, Berlin; New York (1992)
30. Halevi, S., Shoup, V.: Faster homomorphic linear transformations in HElib. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp.

93–120. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/978-3-319-96884-1_4
31. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 364–390. Springer, Cham, Switzerland, San Francisco, CA, USA (Feb 24–28, 2020). https://doi.org/10.1007/978-3-030-40186-3_16
32. Jiang, X., Kim, M., Lauter, K.E., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 1209–1222. ACM Press, Toronto, ON, Canada (Oct 15–19, 2018). https://doi.org/10.1145/3243734.3243837
33. Kim, J., Seo, J., Song, Y.: Simpler and faster BFV bootstrapping for arbitrary plaintext modulus from CKKS. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 2535–2546. ACM Press, Salt Lake City, UT, USA (Oct 14–18, 2024). https://doi.org/10.1145/3658644.3670302
34. Lee, J.W., Lee, E., Lee, Y., Kim, Y.S., No, J.S.: High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 618–647. Springer, Cham, Switzerland, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77870-5_22
35. Lee, Y., Lee, J.W., Kim, Y.S., Kim, Y., No, J.S., Kang, H.: High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 551–580. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-06944-4_19
36. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 227–256. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30620-4_8
37. Liu, J., Zhang, L.F.: Privacy-preserving and publicly verifiable matrix multiplication. IEEE Transactions on Services Computing **16**(3), 2059–2071 (2023). https://doi.org/10.1109/TSC.2022.3215499
38. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer Berlin Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010). https://doi.org/10.1007/978-3-642-13190-5_1
39. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer Berlin Heidelberg, Germany, Athens, Greece (May 26–30, 2013). https://doi.org/10.1007/978-3-642-38348-9_3
40. Moon, J., Yoo, D., Jiang, X., Kim, M.: THOR: Secure transformer inference with homomorphic encryption. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 3765–3779. ACM Press, Taipei, Taiwan (Oct 13–17, 2025). https://doi.org/10.1145/3719027.3765150
41. Park, J.H.: Ciphertext-ciphertext matrix multiplication: Fast for large matrices. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VIII. LNCS, vol. 15608, pp. 153–180. Springer, Cham, Switzerland, Madrid, Spain (May 4–8, 2025). https://doi.org/10.1007/978-3-031-91101-9_6
42. Rader, C.: Discrete Fourier transforms when the number of data samples is prime. Proceedings of the IEEE **56**(6), 1107–1108 (1968). https://doi.org/10.1109/PROC.1968.6477

Appendix

A Revisiting Ciphertext-Ciphertext Matrix Multiplications

Theorem 9 (Homomorphic Trace (ct–ct) [27]). *Let $(b,a),(\beta,\alpha) \in R_q'^2$ be ciphertexts encrypting plaintexts $m,\mu \in R_q'$ under the same secret key $s(X,W)$. Define polynomials in $R_q'[Z]/\langle \Phi_p(Z)\rangle$ as:*

$$d_0 = b(X,Z,W)\cdot\overline{\beta}(Y^{-1},Z^{-1},W^{-1}), \quad d_1 = a(X,Z,W)\cdot\overline{\beta}(Y^{-1},Z^{-1},W^{-1}),$$
$$d_2 = b(X,Z,W)\cdot\overline{\alpha}(Y^{-1},Z^{-1},W^{-1}), \quad d_3 = a(X,Z,W)\cdot\overline{\alpha}(Y^{-1},Z^{-1},W^{-1}).$$

Then, $(\mathrm{Tr}_Z(d_0),\mathrm{Tr}_Z(d_1),\mathrm{Tr}_Z(d_2),\mathrm{Tr}_Z(d_4)) \in R_q'^4$ is a ciphertext encrypting

$$\mathrm{Tr}_Z\left(m(X,Z,W)\cdot\overline{\mu}(Y^{-1},Z^{-1},W^{-1})\right)$$

being decrypted as follows using the secret key $s(X,W)$ and its permutation:

$$\mathrm{Tr}_Z(d_0)+\mathrm{Tr}_Z(d_1)\cdot s(X,W)+\mathrm{Tr}_Z(d_2)\cdot\overline{s}(Y^{-1},W^{-1})+\mathrm{Tr}_Z(d_4)\cdot s(X,W)\overline{s}(Y^{-1},W^{-1}).$$

Proof. For brevity, we let $\mu^* = \overline{\mu}(Y^{-1},Z^{-1},W^{-1})$ (resp. β^*,α^*) and $s^* = \overline{s}(Y^{-1},W^{-1})$ in be the transformed polynomial; similarly, $m' = m(X,Z,W)$ (resp. b',a'). Substituting $m = b + a\cdot s$ and $\mu = \beta + \alpha\cdot s$, we have:

$$\begin{aligned}\mathrm{Tr}_Z\left(m'\cdot\mu^*\right) &= \mathrm{Tr}_Z\left((b'+a'\cdot s)\cdot(\beta^*+\alpha^*\cdot s^*)\right)\\ &= \mathrm{Tr}_Z\left(b'\cdot\beta^* + a'\cdot s\cdot\beta^* + b'\cdot\alpha^*\cdot s^* + a'\cdot s\cdot\alpha^*\cdot s^*\right)\end{aligned}$$

As pointed out in the proof of Lemma 2, Tr_Z is R'-linear, and $s,s^*\in R'$. Therefore, we can factor them out of the trace operation:

$$\begin{aligned}\mathrm{Tr}_Z\left(m'\cdot\mu^*\right) &= \mathrm{Tr}_Z\left(b'\cdot\beta^*\right) + \mathrm{Tr}_Z\left(a'\cdot\beta^*\right)\cdot s + \mathrm{Tr}_Z\left(b'\cdot\alpha^*\right)\cdot s^* + \mathrm{Tr}_Z\left(a'\cdot\alpha^*\right)\cdot s\cdot s^*\\ &= \mathrm{Tr}_Z(d_0) + \mathrm{Tr}_Z(d_1)\cdot s + \mathrm{Tr}_Z(d_2)\cdot s^* + \mathrm{Tr}_Z(d_4)\cdot s\cdot s^*. \qquad \square\end{aligned}$$

Corollary 5 (restated). *Let $(b,a),(\beta,\alpha)\in(R_q^{\mathsf{aux}})^2$ be ciphertexts encrypting plaintexts $m,\mu\in R^{\mathsf{aux}}$ under the same secret key $s(X,W)$. Define polynomials in $R_q^{\mathsf{aux}}[U]/\langle\Phi_p(U)\rangle$ as:*

$$d_0 = b(X,W,U)\cdot\overline{\beta}(X^{-1},V^{-1},U^{-1}), \quad d_1 = a(X,W,U)\cdot\overline{\beta}(X^{-1},V^{-1},U^{-1}),$$
$$d_2 = b(X,W,U)\cdot\overline{\alpha}(X^{-1},V^{-1},U^{-1}), \quad d_3 = a(X,W,U)\cdot\overline{\alpha}(X^{-1},V^{-1},U^{-1}).$$

Then, $(\mathrm{Tr}_U(d_0),\mathrm{Tr}_U(d_1),\mathrm{Tr}_U(d_2),\mathrm{Tr}_U(d_3))\in(R_q^{\mathsf{aux}})^4$ is a ciphertext encrypting

$$\mathrm{Tr}_U\left(m(X,W,U)\cdot\overline{\mu}(X^{-1},V^{-1},U^{-1})\right)$$

being decrypted as follows using the secret key $s(X,W)$ and its permutation:

$$\mathrm{Tr}_U(d_0)+\mathrm{Tr}_U(d_1)\cdot s(X,W)+\mathrm{Tr}_U(d_2)\cdot\overline{s}(X^{-1},V^{-1})+\mathrm{Tr}_U(d_4)\cdot s(X,W)\overline{s}(X^{-1},V^{-1}).$$

Proof. The proof is identical to that of Theorem 9, by replacing Tr_Z with Tr_U and working over R_q^{aux}. The map $\mathrm{Tr}_U : R_q^{\mathsf{aux}}[U]/\langle \Phi_p(U)\rangle \to R_q^{\mathsf{aux}}$ is also R_q^{aux}-linear, and the secret key $s(X, W)$ and its permutation $\overline{s}(X^{-1}, V^{-1})$ are in R^{aux}. □

B Proof of Lemmas

Lemma 1 (restated). *Let $\mathcal{R}$ be a commutative ring containing an element $i \in \mathcal{R}$ with $i^2 = -1$, and let n be a power of two such that $n \in \mathcal{R}^*$, i.e., n is invertible in $\mathcal{R}$. Set $\mathcal{S} := \mathcal{R}[Z]/\langle Z^n - i\rangle$. Then, the group of power automorphisms is isomorphic to the index set*

$$G_{\mathrm{pow}} \cong I_{\mathrm{pow}} := \{k \in \mathbb{Z}_{4n}^* \mid k \equiv 1 \pmod 4\},$$

and the trace map satisfies $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$. Specifically, $\mathrm{Tr}_Z(a) = n \cdot [Z^0]a$. Since $n \in \mathcal{R}^*$, the trace does not vanish on non-zero constants.

Proof. **1. Verification of Automorphisms:** We verify the structure of G_{pow} via the index set I_{pow}.

- **Existence (Well-definedness):** For any $k \in I_{\mathrm{pow}}$ (i.e., $k \equiv 1 \pmod 4$), we have $i^k = i$. Thus the substitution $\sigma_k : Z \mapsto Z^k$ satisfies
$$(Z^k)^n = (Z^n)^k = i^k = i,$$
so it preserves the defining relation $Z^n - i = 0$ and induces a well-defined $\mathcal{R}$-algebra endomorphism of $\mathcal{S}$. Since n is a power of two, $k \equiv 1 \pmod 4$ implies k is odd and hence, $k \in \mathbb{Z}_{4n}^*$. Therefore σ_k is invertible with inverse $\sigma_{k^{-1}}$, and is an automorphism.
- **Completeness (No extra power maps):** Conversely, for a power map $Z \mapsto Z^k$ to be well-defined, we must have
$$(Z^k)^n = i,$$
which implies $i^k = i$ and hence $k \equiv 1 \pmod 4$ since i has order 4. Furthermore, being an automorphism implies invertibility, so $\gcd(k, 4n) = 1$. Thus, any valid index must belong to I_{pow}.
- **Distinctness:** Identify I_{pow} with the set $\{1, 5, \ldots, 4n-3\}$. Assume $\sigma_j = \sigma_k$ for distinct $j, k \in I_{\mathrm{pow}}$. Then $Z^j = Z^k$ in S, hence $Z^{j-k} = 1$; without loss of generality, $m := j - k > 0$, so $0 < m < 4n$. Write $m = qn + r$ with $0 \le r < n$ and $0 \le q < 4$. Using $Z^n = i$ we obtain
$$Z^m = Z^{qn+r} = (Z^n)^q Z^r = i^q Z^r,$$
so $i^q Z^r - 1 = 0$ in $\mathcal{S}$.
Since $f = Z^n - i$ is monic, every element of $\mathcal{S}$ has a unique representative of degree $< n$, so $\{1, Z, \ldots, Z^{n-1}\}$ is an $\mathcal{R}$-basis of $\mathcal{S}$.[7] Therefore $i^q Z^r - 1 = 0$ forces $r = 0$ and $i^q = 1$. As i has order 4, we must have $q \equiv 0 \pmod 4$, hence $q = 0$. Thus $m = 0$, contradicting $m > 0$.

[7] For $g \in \mathcal{R}[Z]$, there exists unique q and r such that $g = q \cdot f + r$ and $\deg(r) < \deg(f)$.

Combining these, the group G_{pow} corresponds exactly to the indices in I_{pow}.

2. Verification of Trace Image: Since every $\sigma \in G_{\mathrm{pow}}$ is an $\mathcal{R}$-algebra automorphism (fixing $\mathcal{R}$ pointwise), the trace map is $\mathcal{R}$-linear. Consequently, we have $\mathrm{Tr}_Z(a) = \sum_{j=0}^{n-1} a_j \,\mathrm{Tr}_Z(Z^j)$ for $a(Z) = \sum_{j=0}^{n-1} a_j Z^j \in \mathcal{S}$, where the trace of basis elements is summed over I_{pow}:

$$\mathrm{Tr}_Z(Z^j) = \sum_{k \in I_{\mathrm{pow}}} (Z^j)^k.$$

- For $j = 0$, we have $\mathrm{Tr}_Z(1) = |I_{\mathrm{pow}}| = n$.
- For $1 \le j \le n-1$, set $r := Z^{4j}$. Using representatives $k = 1 + 4t$ ($t = 0, \ldots, n-1$),

$$\mathrm{Tr}_Z(Z^j) = \sum_{t=0}^{n-1} Z^{j(1+4t)} = Z^j \sum_{t=0}^{n-1} r^t.$$

Since $n = 2^m$, we use the factorization

$$\sum_{t=0}^{n-1} r^t = \prod_{s=0}^{m-1} \left(1 + r^{2^s}\right),$$

which follows from repeatedly applying

$$\sum_{t=0}^{2^{s+1}-1} r^t = \left(\sum_{t=0}^{2^s-1} r^t\right)\left(1 + r^{2^s}\right).$$

Now $Z^n = i$ implies $Z^{2n} = i^2 = -1$. Write $j = 2^u v$ with v odd and $0 \le u < m$, and choose $s := m - 1 - u$. Then

$$r^{2^s} = (Z^{4j})^{2^s} = Z^{4j \cdot 2^s} = Z^{4(2^u v)2^{m-1-u}} = Z^{v \cdot 2^{m+1}} = Z^{v \cdot 2n} = (Z^{2n})^v = (-1)^v = -1.$$

Hence the corresponding factor is

$$1 + r^{2^s} = 1 + (-1) = 0,$$

so the entire product vanishes, and therefore $\mathrm{Tr}_Z(Z^j) = 0$.

Hence

$$\mathrm{Tr}_Z(a) = n \cdot a_0 \in \mathcal{R},$$

and therefore $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$. □

Lemma 2 (restated). *Let $p \in \mathcal{R}^*$ be an odd prime. Set $\mathcal{S} := \mathcal{R}[Z]/\langle \Phi_p(Z) \rangle$. Then, $G_{\mathrm{pow}} \cong \mathbb{Z}_p^*$, and*

$$\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}.$$

Specifically, $\mathrm{Tr}_Z(a) = p \cdot a_0 - \sum_{j=0}^{\varphi(p)-1} a_j$, where $a = \sum_{j=0}^{\varphi(p)-1} a_j Z^j$.

Proof. **1. Verification of Automorphisms:** We establish the isomorphism $G_{\mathrm{pow}} \cong \mathbb{Z}_p^*$.

– **Existence:** For any $k \in \mathbb{Z}_p^*$, the substitution $\sigma_k : Z \mapsto Z^k$ preserves the defining relation $\Phi_p(Z) = 0$ in $\mathcal{S}$. Therefore, it induces a well-defined $\mathcal{R}$-algebra endomorphism σ_k on $\mathcal{S}$. Moreover, σ_k is invertible with inverse $\sigma_{k^{-1}}$, since k^{-1} exists modulo p.
– **Completeness:** If $\gcd(k, p) \neq 1$, then $p \mid k$, say $k = mp$. Since $Z^p = 1$ in $\mathcal{S}$, we have $Z^k = (Z^p)^m = 1$ and hence

$$\Phi_p(Z^k) = \Phi_p(1) = p \neq 0,$$

contradicting well-definedness. Therefore $\gcd(k, p) = 1$.
– **Distinctness:** Identify $I_{\text{pow}} = (\mathbb{Z}/p\mathbb{Z})^*$ with the set of representatives $\{1, \ldots, p-1\}$. Assume, for two distinct $j, k \in I_{\text{pow}}$, that $\sigma_j = \sigma_k$. Then $Z^j = Z^k$ in $\mathcal{S}$, which implies $Z^{j-k} = 1$ in $\mathcal{S}$; without loss of generality, assume $j > k$. Thus $\Phi_p(Z) \mid (Z^{j-k} - 1)$. This is a contradiction, as $\Phi_p(Z)$ is monic and $\deg(Z^{j-k} - 1) < \deg(\Phi_p)$.

Combining these, the set of power automorphisms is indexed by $\mathbb{Z}_p^*$.

2. Verification of Trace Image: By definition of the power-trace, we have for any $a(Z) = \sum_{j=0}^{\varphi(p)-1} a_j Z^j \in \mathcal{S}$ for $\deg(\Phi_p) = \varphi(p) = p - 1$,

$$\text{Tr}_Z(a) = \sum_{k \in \mathbb{Z}_p^*} a(Z^k) = \sum_{j=0}^{\varphi(p)-1} a_j \left(\sum_{k \in \mathbb{Z}_p^*} (Z^j)^k \right).$$

– For $j = 0$, we have $\sum_{k \in \mathbb{Z}_p^*} Z^0 = p - 1$.
– Otherwise, we have $j \in \mathbb{Z}_p^*$, hence multiplication by j permutes $\mathbb{Z}_p^*$, and thus

$$\sum_{k \in \mathbb{Z}_p^*} Z^{jk} = \sum_{r \in \mathbb{Z}_p^*} Z^r = \sum_{r=1}^{p-1} Z^r = \Phi_p(Z) - 1 = -1.$$

Finally, we have

$$\text{Tr}_Z(a) = p \cdot a_0 - \sum_{j=0}^{\varphi(p)-1} a_j,$$

implying $\text{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$ since $a_j \in \mathcal{R}$. □

Lemma 3 (restated). *Let $\mathcal{R}$ be a commutative ring and let p be an odd prime with $p \in \mathcal{R}^*$. Fix $k \geq 1$ and set $\mathcal{S} := \mathcal{R}[Z]/\langle \Phi_{p^k}(Z) \rangle$. Then $G_{\text{pow}} \cong \mathbb{Z}_{p^k}^*$ and $\text{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$. More precisely, writing $a = \sum_{j=0}^{\varphi(p^k)-1} a_j Z^j$ (with $\deg < \varphi(p^k)$), we have*

$$\text{Tr}_Z(a) = p^k \cdot a_0 - p^{k-1} \textstyle\sum_{t=0}^{p-2} a_{tp^{k-1}} \in \mathcal{R}.$$

Proof. **1. Verification of automorphisms.** Let $u \in \mathbb{Z}$ and define σ_u by $\sigma_u(Z) = Z^u$. Using $\Phi_{p^k}(X) = \Phi_p(X^{p^{k-1}}) = \sum_{t=0}^{p-1} X^{tp^{k-1}}$, set $Y := Z^{p^{k-1}}$ so that

$$\Phi_p(Y) = 1 + Y + \cdots + Y^{p-1} = 0 \quad \text{in } \mathcal{S}.$$

If $\gcd(u,p)=1$, then

$$\Phi_{p^k}(Z^u)=\Phi_p(Y^u)=\sum_{t=0}^{p-1}Y^{ut}=\sum_{t=0}^{p-1}Y^t=\Phi_p(Y)=0,$$

since multiplication by u permutes $\{0,\dots,p-1\}$ modulo p; hence σ_u is well-defined. Also $(Z^u)^{p^k}=1$ holds in $\mathcal{S}$, so choosing v with $uv\equiv 1 \pmod{p^k}$ gives $\sigma_v\circ\sigma_u=\mathbf{1}$, thus σ_u is an automorphism.

Conversely, if $p\mid u$, then $Y^u=1$ and

$$\Phi_{p^k}(Z^u)=\Phi_p(1)=p\neq 0,$$

contradicting well-definedness; hence any power automorphism must satisfy $\gcd(u,p)=1$, i.e., $u\in\mathbb{Z}_{p^k}^*$. Finally, if $\sigma_u=\sigma_v$ then $Z^{u-v}=1$, which forces $u\equiv v \pmod{p^k}$, so the indexing is by $\mathbb{Z}_{p^k}^*$. Therefore $G_{\mathrm{pow}}\cong\mathbb{Z}_{p^k}^*$.

2. Verification of trace image. Write $a=\sum_{i=0}^{\varphi(p^k)-1}a_iZ^i$ in $\mathcal{S}$ (unique with $\deg<\varphi(p^k)$). By $\mathcal{R}$-linearity,

$$\mathrm{Tr}_Z(a)=\sum_{i=0}^{\varphi(p^k)-1}a_i\,\mathrm{Tr}_Z(Z^i),\qquad \mathrm{Tr}_Z(Z^i)=\sum_{u\in\mathbb{Z}_{p^k}^*}Z^{ui}.$$

Set $Y:=Z^{p^{k-1}}$. Since $\Phi_{p^k}(Z)=\Phi_p(Z^{p^{k-1}})$, we have in $\mathcal{S}$

$$\Phi_p(Y)=1+Y+\cdots+Y^{p-1}=0.$$

First, for $i=0$,

$$\mathrm{Tr}_Z(1)=|\mathbb{Z}_{p^k}^*|=\varphi(p^k)=p^{k-1}(p-1).$$

Now assume $i\geq 1$. Every $u\in\mathbb{Z}_{p^k}^*$ can be written uniquely as $u=u_0+tp^{k-1}$ with $u_0\in\mathbb{Z}_{p^{k-1}}^*$ and $t\in\{0,\dots,p-1\}$, hence

$$\begin{aligned}\mathrm{Tr}_Z(Z^i)&=\sum_{u_0\in\mathbb{Z}_{p^{k-1}}^*}\sum_{t=0}^{p-1}Z^{(u_0+tp^{k-1})i}=\sum_{u_0\in\mathbb{Z}_{p^{k-1}}^*}Z^{u_0i}\sum_{t=0}^{p-1}(Z^{p^{k-1}i})^t\\&=\sum_{u_0\in\mathbb{Z}_{p^{k-1}}^*}Z^{u_0i}\sum_{t=0}^{p-1}(Y^i)^t.\end{aligned}$$

If $p\nmid i$, then $Y^i\neq 1$, so $\sum_{t=0}^{p-1}(Y^i)^t=0$ and therefore $\mathrm{Tr}_Z(Z^i)=0$. If $p\mid i$, then $Y^i=1$ and the inner sum equals p, giving

$$\mathrm{Tr}_Z(Z^i)=p\sum_{u_0\in\mathbb{Z}_{p^{k-1}}^*}Z^{u_0i}.$$

Iterating this reduction $k-1$ times shows that $\mathrm{Tr}_Z(Z^i) = 0$ unless $p^{k-1} \mid i$.

Thus it remains to consider $i = p^{k-1}t$ with $t \in \{0, \dots, p-2\}$. Then

$$\mathrm{Tr}_Z(Z^{p^{k-1}t}) = \sum_{u \in \mathbb{Z}_{p^k}^*} (Z^{p^{k-1}})^{ut} = \sum_{u \in \mathbb{Z}_{p^k}^*} Y^{ut} = p^{k-1} \sum_{r=1}^{p-1} Y^{rt}.$$

For $t = 0$ this equals $p^{k-1}(p-1)$. For $1 \le t \le p-2$, we have $t \in \mathbb{Z}_p^*$, so $r \mapsto rt$ permutes $\{1, \dots, p-1\}$ and hence

$$\sum_{r=1}^{p-1} Y^{rt} = \sum_{r=1}^{p-1} Y^r = -1$$

(using $1 + \sum_{r=1}^{p-1} Y^r = 0$). Therefore $\mathrm{Tr}_Z(Z^{p^{k-1}t}) = -p^{k-1}$.

Consequently,

$$\mathrm{Tr}_Z(a) = a_0 \cdot p^{k-1}(p-1) + \sum_{t=1}^{p-2} a_{tp^{k-1}} \cdot (-p^{k-1}) = p^k \cdot a_0 - p^{k-1} \sum_{t=0}^{p-2} a_{tp^{k-1}},$$

so $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$. □

Lemma 4 (restated). *Let $\psi : \mathcal{R} \to \mathcal{R}'$ be a ring homomorphism and let $f(Z) \in \mathcal{R}[Z]$ be monic. Set*

$$\mathcal{S} := \mathcal{R}[Z]/\langle f(Z) \rangle, \qquad \mathcal{S}' := \mathcal{R}'[Z]/\langle \psi(f)(Z) \rangle,$$

and let $\psi_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}'$ be the induced homomorphism on coefficients. Then, for all $a \in \mathcal{S}$,

$$\psi_{\mathcal{S}}\big(\mathrm{Tr}_Z(a)\big) = \mathrm{Tr}_Z\big(\psi_{\mathcal{S}}(a)\big),$$

where Tr_Z on $\mathcal{S}'$ is defined as in Definition 2. Moreover, if $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$, then

$$\psi\big(\mathrm{Tr}_Z(a)\big) = \mathrm{Tr}_Z\big(\psi_{\mathcal{S}}(a)\big).$$

Proof. It suffices to show that $\psi_{\mathcal{S}}$ commutes with every power automorphism. Let $\sigma \in G_{\mathrm{pow}}$ be the automorphism defined by $Z \mapsto Z^k$. Then the same power substitution $Z \mapsto Z^k$ defines an automorphism on both $\mathcal{S}$ and $\mathcal{S}'$.

Since $\psi_{\mathcal{S}}$ acts only on the coefficients and σ acts only on the variable exponent, they act independently. Specifically, for any polynomial representative $g(Z) = \sum g_j Z^j \in \mathcal{R}[Z]$, we have:

$$\psi_{\mathcal{S}}\big(\sigma(g(Z))\big) = \psi_{\mathcal{S}}\big(\textstyle\sum_j g_j Z^{kj}\big) = \sum_j \psi(g_j) Z^{kj} = \sigma\big(\textstyle\sum_j \psi(g_j) Z^j\big) = \sigma\big(\psi_{\mathcal{S}}(g(Z))\big).$$

Thus, $\psi_{\mathcal{S}} \circ \sigma = \sigma \circ \psi_{\mathcal{S}}$.

By the linearity of $\psi_{\mathcal{S}}$, applying this to the trace definition yields:

$$\begin{aligned}\psi_{\mathcal{S}}\big(\mathrm{Tr}_Z(a)\big) &= \psi_{\mathcal{S}}\big(\textstyle\sum_{\sigma \in G_{\mathrm{pow}}} \sigma(a)\big) = \sum_{\sigma \in G_{\mathrm{pow}}} \psi_{\mathcal{S}}(\sigma(a)) \\ &= \textstyle\sum_{\sigma \in G_{\mathrm{pow}}} \sigma(\psi_{\mathcal{S}}(a)) = \mathrm{Tr}_Z\big(\psi_{\mathcal{S}}(a)\big).\end{aligned}$$

Finally, if $\mathrm{Tr}_Z(a) \in \mathcal{R}$, then $\psi_{\mathcal{S}}$ coincides with ψ on this element, yielding the second identity. □

Theorem 5 (restated). *Let $a(X,W,V), b(X,W,V)$ be polynomials in R^{aux} defined by prime p, and define the involution $b^*(X,W,V) := \overline{b}(X^{-1},W^{-1},V^{-1})$. Define matrices $\hat{A}$ and $\hat{B}'$ in $\mathcal{M}_{\varphi(p)}(R)$ by arranging the W,V-coefficients (with coefficients in $\mathbb{Z}[i][X]/\langle X^n - i\rangle$) as follows:*

$$[\hat{A}]_{j\ell} = [W^j V^\ell]a, \qquad [\hat{B}']_{k\ell} = [W^k V^{-\ell}]b^*,$$

where the entries are in $\mathbb{Z}[i][X]/\langle X^n - i\rangle$.

Then, for $c = \mathrm{Tr}_U\left(a(X,W,U)\cdot b^*(X,V,U)\right)$, the coefficient of W^jV^k is given by:

$$[W^jV^k]c = p\cdot[\hat{C}]_{jk} - [\hat{A}\mathbf{1}]_j\cdot[\hat{B}'\mathbf{1}]_k = [p\cdot\hat{C} - (\hat{A}\mathbf{1})\cdot(\hat{B}'\mathbf{1})^{\mathrm{T}}]_{jk},$$

where $\hat{C} = \hat{A}\hat{B}'^{\mathrm{T}}$ and $\mathbf{1}$ is the all-one vector of length $\varphi(p)$.

Proof. Let $d(X,W,V,U) = a(X,W,U)\cdot b^*(X,V,U) \in R_q^{\mathsf{aux}}[U]/\langle\Phi_p(U)\rangle$, and $d = \sum_{\ell=0}^{\varphi(p)-1} d_\ell U^\ell$. Applying Lemma 2 for $R^{\mathsf{aux}}[U]/\langle\Phi_p(U)\rangle$ we have:

$$\mathrm{Tr}_U(d) = p\cdot d_0 - \textstyle\sum_{\ell\in[0,\varphi(p))} d_\ell.$$

Let us denote the coefficients of a and b^* explicitly as:

$$a(X,W,U) = \textstyle\sum_{j,\alpha} a_{j,\alpha}(X)W^jU^\alpha, \quad b^*(X,V,U) = \sum_{k,\beta} b^*_{k,\beta}(X)V^kU^\beta.$$

Similarly to the proof of Theorem 3, the constant term d_0 is given by:

$$d_0 = \textstyle\sum_{j,k}\sum_\alpha a_{j,\alpha}\cdot b^*_{k,-\alpha}\cdot W^jV^k.$$

For a fixed (j,k) term, we have:

$$[W^jV^k]d_0 = \sum_{\alpha=0}^{\varphi(p)-1} [\hat{A}]_{j\alpha}\cdot[\hat{B}']_{k\alpha}.$$

For the remaining term, we consider $a(X,W,U)\cdot b^*(X,V,U)$, where $b^*(X,V,U)$ is represented as a polynomial in U^{-1}, in other words,

$$a(X,W,U) = \textstyle\sum_{\nu\in[0,\varphi(p))}([V^\nu]a)\cdot U^\nu \text{ and } b^*(X,V,U) = \sum_{\nu\in[0,\varphi(p))}([V^{-\nu}]b^*)\cdot U^{-\nu}.$$

Then, $d = a\cdot b^*$ is uniquely represented as $d = \sum_{\ell=-\varphi(p)+1}^{\varphi(p)-1}\tilde{d}_\ell U^\ell$, where $\tilde{d}_\ell = \sum_{\alpha-\beta=\ell}([V^\alpha]a)\cdot([V^{-\beta}]b^*)$. Since we have $U^p = 1$, the following holds:

$$d = \tilde{d}_0 + \sum_{\ell=1}^{\varphi(p)-1}(\tilde{d}_\ell + \tilde{d}_{\ell-p})U^\ell.$$

$\sum_{\ell\in[0,\varphi(p))} d_\ell$ is thus equivalent to $\sum_{\ell=-\varphi(p)+1}^{\varphi(p)-1}\tilde{d}_\ell$. In other words,

$$\begin{aligned}\textstyle\sum_{\ell\in[0,\varphi(p))} d_\ell &= \textstyle\sum_{\ell=-\varphi(p)+1}^{\varphi(p)-1}\sum_{\alpha-\beta=\ell}([V^\alpha]a)\cdot([V^{-\beta}]b^*)\\ &= \textstyle\sum_\alpha^{\varphi(p)-1}[V^\alpha]a\cdot\sum_\beta^{\varphi(p)-1}[V^{-\beta}]b^*.\end{aligned}$$

Thus, for a fixed index (j,k), we have:

$$\begin{aligned}[W^jV^k]\sum_{\ell\in[0,\varphi(p))} d_\ell &= \sum_{\alpha}^{\varphi(p)-1}[W^jU^\alpha]a\cdot\sum_{\beta}^{\varphi(p)-1}[V^kU^{-\beta}]b^* \\ &= \sum_{\alpha=0}^{\varphi(p)-1}[\hat{A}]_{j\alpha}\cdot\sum_{\beta=0}^{\varphi(p)-1}[\hat{B'}]_{k\beta} = [\hat{A}\mathbf{1}]_j\cdot[\hat{B'}\mathbf{1}]_k.\end{aligned}$$

Combining these results, the claim follows. □

Lemma 5 (restated). *Let $f \in \mathcal{R}[Z]$ be monic and set $S := \mathcal{R}[Z]/\langle f(Z)\rangle$. Assume that:*

1. *the power-trace is $\mathcal{R}$-valued, i.e., $\mathrm{Tr}_Z(\mathcal{S}) \subseteq \mathcal{R}$;*
2. *f splits in $\mathcal{R}$ with distinct roots $\Omega = \{\zeta^k \mid k \in I_{\mathrm{pow}}\} \subset \mathcal{R}$, where $\zeta \in \Omega$ is fixed and $k \mapsto \zeta^k$ is a bijection $I_{\mathrm{pow}} \to \Omega$.*

Then, for any $a \in \mathcal{S}$, we have $\mathrm{Tr}_Z(a) = \sum_{\omega\in\Omega} a(\omega)$.

Proof. By definition, $\mathrm{Tr}_Z(a) = \sum_{k\in I_{\mathrm{pow}}} a(Z^k) \in \mathcal{R}$, so $\mathrm{Tr}_Z(a)$ is a constant (i.e., Z-independent) element of $\mathcal{R}$. In particular, for any root ζ of f, evaluation at $Z = \zeta$ is well-defined and leaves $\mathrm{Tr}_Z(a)$ unchanged, hence

$$\mathrm{Tr}_Z(a) = (\mathrm{Tr}_Z(a))(\zeta) = \sum_{k\in I_{\mathrm{pow}}} a(\zeta^k).$$

By Assumption (2), as k ranges over I_{pow}, the values ζ^k enumerate the set Ω exactly once. Therefore,

$$\sum_{k\in I_{\mathrm{pow}}} a(\zeta^k) = \sum_{\omega\in\Omega} a(\omega),$$

and the claim follows. □

Lemma 6 (restated). *Given polynomials m, μ in $R'[Z]/\langle Z^n - i\rangle$, assuming the use of the powerful basis, we have*

$$\|\mathrm{Tr}_Z(m\cdot\mu)\|_\infty \le n\cdot\|m\|_\infty^{\mathsf{can}}\cdot\|\mu\|_\infty^{\mathsf{can}},$$

where $\|\cdot\|_\infty^{\mathsf{can}}$ denotes the infinity norm of the canonical embedding.

Proof. The trace operation $\mathrm{Tr}_Z(\cdot)$ corresponds to the sum of power automorphisms which is a norm-preserving map. Hence,

$$\|\mathrm{Tr}_Z(m\cdot\mu)\|_\infty \le \sum_{\sigma\in G_{pow}}\|\sigma(m\cdot\mu)\|_\infty = n\cdot\|m\cdot\mu\|_\infty \le n\cdot\|m\|_\infty^{\mathsf{can}}\cdot\|\mu\|_\infty^{\mathsf{can}}.$$

For the last inequality, we recall that for the powerful basis, the coefficient ℓ_∞ norm is bounded by the canonical embedding ℓ_∞ norm [39]. □

Theorem 7 (restated). *Let $R = \mathbb{Z}[i][X,W]/\langle X^n - i, \Phi_p(W)\rangle$. Consider the partial decoding map*

$$\Psi_{X,W} : R \to \mathbb{C}^{n\times\varphi(p)}, \quad m \mapsto M = [m(\zeta_j,\eta_k)]_{0\le j<n, 0\le k<\varphi(p)}.$$

For a target matrix $A \in \mathcal{M}_n(\mathbb{C})$, let $a(X,Z) \in R[Z]/\langle Z^n - i\rangle$ be a polynomial independent of W, satisfying $a(\zeta_j, \zeta_k) = A_{jk}$ for all $j,k \in [0,n)$. Then, for $c(X,W) = \mathrm{Tr}_Z\left(a(X,Z)\cdot m(Z,W)\right)$, we have

$$\Psi_{X,W}(c) = AM,$$

where Tr_Z denotes the trace over the extension $R[Z]/\langle Z^n - i\rangle$.

Proof. Let Ψ_{jk} be the evaluation homomorphism $m \mapsto m(\zeta_j, \eta_k)$. By applying Lemmas 4 and 5, and noting that Tr_Z corresponds to the sum over the roots ζ_ℓ of $Z^n - i$, we obtain:

$$\begin{aligned}
\Psi_{jk}(c) &= \mathrm{Tr}_Z\left(a(\zeta_j, Z)\cdot m(Z,\eta_k)\right) \\
&= \textstyle\sum_{\iota=0}^{n-1} a(\zeta_j,\zeta_\iota)\cdot m(\zeta_\iota,\eta_k) && \text{by Lemma 5} \\
&= \textstyle\sum_{\iota=0}^{n-1} A_{j\iota}\cdot M_{\iota k} = (AM)_{jk}\,. && \square
\end{aligned}$$

Theorem 8 (restated). *Define R and $\Psi_{X,W}$ as in Theorem 7. For a target matrix $A \in \mathcal{M}_{\varphi(p)}(\mathbb{C})$, let $a(W,U) \in \mathbb{Z}_q[i][W,U]/\langle \Phi_p(W), \Phi_p(U)\rangle$ be a polynomial satisfying the interpolation condition: $a(\eta_j,\eta_k) = A_{jk}$ for all $j,k \in [0,\varphi(p))$. Then, for $c(X,W) = \mathrm{Tr}_U\left(a(W,U)\cdot m(X,U)\right)$, we have*

$$\Psi_{X,W}(c) = MA^{\mathrm{T}},$$

where Tr_U denotes the trace over the extension $R[U]/\langle \Phi_p(U)\rangle$.

Proof. Analogous to the proof of Theorem 7, we obtain:

$$\begin{aligned}
\Psi_{jk}(c) &= \mathrm{Tr}_U\left(a(\eta_k, U)\cdot m(\zeta_j,U)\right) \\
&= \textstyle\sum_{\iota=0}^{\varphi(p)-1} a(\eta_k,\eta_\iota)\cdot m(\zeta_j,\eta_\iota) && \text{by Lemma 5} \\
&= \textstyle\sum_{\iota=0}^{\varphi(p)-1} A_{k\iota}\cdot M_{j\iota} = \left(MA^{\mathrm{T}}\right)_{jk}\,. && \square
\end{aligned}$$

C Algorithms

For $a \in R'$, let $a_{j,k,\ell}$ denote the coefficient of $X^j Y^k W^\ell$. Similarly, for $b \in R^{\mathsf{aux}}$, let $b_{j,k,\ell}$ denote the coefficient of $X^j W^k V^\ell$.

Algorithm 2 Matrix Multiplication over R' (Power-of-two)

Input: $a, b \in R'$ where $a(\zeta_j, \zeta_k, \eta_\ell) = A^{(\ell)}_{jk}$ and $b(\zeta_j, \zeta_k, \eta_\ell) = B^{(\ell)}_{jk}$.
Output: $c \in R'$ where $c(\zeta_j, \zeta_k, \eta_\ell) = \left[A^{(\ell)} B^{(\ell)\mathrm{H}}\right]_{jk}$.

$b^* \leftarrow \mathsf{Herm}(b)$ ▷ Hermitian transpose with (X, Y)
$\tilde{a} \leftarrow \mathrm{NTT}_W(a), \tilde{b}^* \leftarrow \mathrm{NTT}_W(b^*)$ ▷ W-axis NTT
for $\ell = 0$ to $\varphi(p) - 1$ **do**
  for $j = 0$ to $n - 1$ **do**
    for $k = 0$ to $n - 1$ **do**
      $\tilde{c}_{j,k,\ell} = n \cdot \sum_{\alpha=0}^{n-1} \tilde{a}_{j,\alpha,\ell} \cdot \tilde{b}^*_{n-\alpha,k,\ell}$ ▷ Indices are taken modulo n
$c \leftarrow \mathrm{INTT}_W(\tilde{c})$ ▷ W-axis INTT
return c

Algorithm 3 Matrix Multiplication over R^{aux} (Non-power-of-two)

Input: $a, b \in R^{\mathsf{aux}}$ where $a(\zeta_\ell, \eta_j, \eta_k) = A^{(\ell)}_{jk}$ and $b(\zeta_\ell, \eta_j, \eta_k) = B^{(\ell)}_{jk}$.
Output: $c \in R^{\mathsf{aux}}$ where $c(\zeta_\ell, \eta_j, \eta_k) = \left[A^{(\ell)} B^{(\ell)\mathrm{H}}\right]_{jk}$.

$b^* \leftarrow \mathsf{Herm}(b)$ ▷ Hermitian transpose with (W, V)
$\tilde{a} \leftarrow \mathrm{NTT}_X(a), \tilde{b}^* \leftarrow \mathrm{NTT}_X(b^*)$ ▷ X-axis NTT
for $\ell = 0$ to $n - 1$ **do**
  for $j = 0$ to $\varphi(p) - 1$ **do**
    for $k = 0$ to $\varphi(p) - 1$ **do**
      $d_{\ell,j,k} = \sum_{\alpha=0}^{\varphi(p)-1} \tilde{a}_{\ell,j,\alpha} \cdot \sum_{\alpha=0}^{\varphi(p)-1} \tilde{b}^*_{\ell,\alpha,k}$
      $\tilde{c}_{\ell,j,k} = p \cdot \sum_{\alpha=0}^{\varphi(p)-1} \tilde{a}_{\ell,j,\alpha} \cdot \tilde{b}^*_{\ell,\varphi(p)-\alpha,k} - d_{\ell,j,k}$
      ▷ Indices are taken modulo $\varphi(p)$
$c \leftarrow \mathrm{INTT}_X(\tilde{c})$ ▷ X-axis INTT
return c